Méthodes numériques de représentation à variables séparées pour la résolution des problèmes paramétriques en mécanique non-linéaire des structures

Sophie Cartel

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n°432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité

« Mécanique »

présentée et soutenue publiquement par

Sophie CARTEL

le 25 novembre 2011

Méthodes numériques de représentation à variables séparées pour la résolution des problèmes paramétriques en mécanique non-linéaire des structures

Directeur de thèse : **David RYCKELYNCK**

THÈSE

Jury

M. Florian DE VUYST, Professeur des Universités, CMLA , ENS CACHAN	Rapporteur
M. Aziz HAMDOUNI, Professeur des Universités, LEPTIAB , Université de la Rochelle	Président
M. Pierre JOYOT, Maître de Conférence HDR, ESTIA Recherche, ESTIA	Rapporteur
M. David RYCKELYNCK, Maître de Recherche, Centre des Matériaux, Mines ParisTech	Directeur

MINES ParisTech
Centre des Matériaux CNRS UMR 7633
B.P. 87, 91003 EVRY Cedex, France

Remerciements

Voilà pour moi l'occasion de faire mes remerciements en bonne et due forme, puisque le jour J, la gorge nouée, je n'ai su trouver les mots pour remercier comme il se doit les personnes qui ont été les plus présentes pour moi. Mais comme on dit, les paroles s'envolent et les écrits restent...

Je tiens donc à remercier en premier lieu le directeur de cette thèse, David Ryckelynck, pour m'avoir fait confiance malgré mes connaissances plutôt légères en mécanique et en science des matériaux. Ses conseils, sa disponibilité, et son soutien ont contribué à la réalisation de ce travail.

J'exprime mes sincères remerciements à tous les membres du jury qui ont bien voulu juger ce travail. Je remercie donc M. Aziz Hamdouni, Professeur à l'Université de la Rochelle, qui m'a fait l'honneur de présider ce jury. Je remercie M. Florian De Vuyst, Professeur à l'ENS Cachan, et M. Pierre Joyot, Maître de conférence HDR à l'ESTIA, d'avoir bien voulu accepter la charge de rapporteur. Tous vos conseils et vos remarques m'ont été précieux.

Je tiens à adresser mes remerciements au pôle de compétitivité System@tic, dont le financement a permis à ce projet de thèse de voir le jour.

J'ai passé ces 3 années de thèse au Centre des Matériaux, et ce fut l'occasion de rencontrer et d'échanger avec de nombreuses personnes, aussi bien sur le plan humain que scientifique. Je pense notamment aux nombreuses et riches discussions avec M. André Pineau, et j'ai eu la chance d'avoir sur place M. Gilles Rousselier dont les explications sur son modèle m'ont permis d'avancer.

Je souhaite à présent exprimer toute ma gratitude à Djamel Missoum-Benziane, avec qui j'ai eu le plaisir de partager mon bureau. Je suis consciente du temps que tu as passé à répondre à mes questions, mais aussi à me déstresser, ce qui, tu en conviendras, n'était pas tâche (grasse...) facile ! Tu en

as d'ailleurs parfois joué, notamment à l'approche de conférences... Je te suis vraiment reconnaissante pour toute l'aide que tu m'as apportée tout au long de ces 3 années passées au centre, mais aussi les samedis et dimanches matins, t'empêchant de faire la grasse (!) matinée. La blonde te dit un énorme MERCI.
Vlad, mon autre colocataire de bureau(x) et photographe de choc, j'ai passé également de супер bons moments avec toi, que ce soit en A113 (haaaaa, la fondue au chocolat...), B109 (les nombreuses réunions de la mafia russe) , ou à l'extérieur. Большое спасибо Влад!

L'ambiance au centre n'aurait pas été la même sans l'équipe de choc Maria, Nicole, Anne et Yves. Deux univers de travail différents (je n'aurai finalement été au MEB que pour voir un pou, une araignée, ou encore un cheveu écaillé), mais les 12H, les repas de Noël, nous auront rapprochés. De très bons moments également en dehors du labo. Merci d'avoir souffert 2H de votre vie pour me soutenir, je vous avais pourtant dit de venir "copeau" ! J'espère que nous nous reverrons vite.
Je tiens également à remercier toutes les personnes, entre autres mes collègues thésards du centre qui se sont déplacés pour ma soutenance, cela m'a fait plaisir, et surtout, sans vous, je n'aurai jamais pu finir le champagne !
J'ai aussi une pensée pour les moments passés avec Fatima, Florence, Sergueï, Bahram...

Je n'oublie pas mon professeur d'anglais Troy Speight, grâce à qui my tailor is rich ! Ma famille et moi même vous adressons nos plus sincères remerciements pour votre présence le jour de la soutenance, et les souvenirs que nous garderons sur vidéo.

Parmi les permanents du centre, je me dois de remercier Olivier et Greg pour leur dépannages informatiques, parfois même à distance.
Merci à Odile, pour les articles souvent réclamés à la dernière minute, merci à Saro, Konaly, Liliane, Anne, aux deux Véronique, pour l'aide que chacune d'entre elles a pu m'apporter dans différents domaines administratifs.
Enfin, Catherine, nos conversations à l'accueil le matin vont beaucoup me manquer.

Je ne peux oublier d'exprimer ma profonde gratitude à Philippe Thomas qui a eu confiance dans mon travail, et grâce à qui j'ai pu intégrer aujourd'hui Dassault Aviation. Merci également à mes supérieurs et collègues actuels qui ont compris mes nombreuses absences avant la soutenance, et qui ont bien voulu jouer les cobayes en assistant à ma répétition.

J'aimerai adresser un merci général à tous mes amis qui ont été là pour les bons et les mauvais moments, à tous ceux qui m'ont adressé leurs encouragements. En effet, chaque petit mot de soutien est important dans les derniers jours de thèse, et cela m'a fait chaud au cœur. Un merci particulier à Seb, "mon papounet", pour tous ses conseils et encouragements d'avant, pendant et après thèse.

Cette partie du mémoire, que je pensais la plus facile à écrire, n'est pas aussi évidente qu'elle n'y parait, et j'espère sincèrement n'avoir oublié personne. Ainsi, à toutes les personnes que j'ai citées et celles que j'aurai pu oublier : MERCI !

Enfin, mes remerciements les plus chaleureux sont pour Stéphane et mes parents. Mes parents, parce qu'ils m'ont toujours donné les moyens de réussir. Les valeurs et principes qu'ils m'ont inculqués me serviront toute ma vie. Leur présence à ma soutenance malgré les kilomètres et la maladie était indispensable. Je ne saurai trouver les mots pour les remercier.
Quant à Stéphane, ne lui parlez surtout plus de réduction de modèle, il en a assez entendu pour des années. Je le remercie pour son soutien inconditionnel tout au long de ces années de thèse, pour sa patience, son amour, pour m'avoir permis de rédiger dans les meilleures conditions. Cette thèse nous a pris un temps précieux, mais une nouvelle vie ne fait que commencer...
Je tenais ainsi à leur exprimer ici toute ma gratitude et mon amour. Cette thèse leur est dédiée.

J'aimerai terminer ces longs remerciements dignes d'une remise d'un oscar, par une citation qui résumera un travail de thèse, mais me rappellera également ma Haute Savoie natale :

> *"Tout le monde veut vivre au sommet de la montagne, sans soupçonner que le vrai bonheur est dans la manière de gravir la pente."*
>
> Gabriel Garcia Marquez

Notations

Vecteurs, matrices et tenseurs

$\{X\}$ Vecteur colonne

$\underline{\boldsymbol{X}}$ Vecteur (tenseur du premier ordre)

$\underset{\sim}{\boldsymbol{X}}$ Tenseur du second ordre

$[X]$ Matrice

Abréviations

POD	Proper Orthogonal Decomposition (Décomposition Orthogonale aux valeurs Propres)
ROM	Reduced Order Model (Réduction d'Ordre de Modèles)
ER	Évènements Réccurents
APR	A Priori Reduction (A Priori Réduction)
APHR	A Priori Hyper Reduction (A Priori Hyper Réduction)
EF	Éléments Finis
RID	Reduced Integration Domain (Domaine d'Intégration Réduit)

Table des matières

Introduction

La simulation numérique s'est largement imposée dans le développement et la mise au point des systèmes complexes. Les simulations concernant un même système au cours de sa conception, de sa fabrication et de son suivi en service, sont nombreuses. La simulation numérique désigne le procédé selon lequel des modèles théoriques permettent de représenter un phénomène physique.
Cette thèse s'inscrit dans le cadre de différents projets proposés par le pôle de compétitivité system@tic. Les objectifs du projet CSDL (Complex System Design Lab) et du projet EHPOC (Environnement Haute Performance pour l'Optimisation et la Conception) sont de disposer d'outils et méthodologies au meilleur niveau pour la conception de systèmes complexes : les produits conçus doivent évidemment être de qualité et robustes, mais également conçus en temps réduits. Les logiciels de modélisation et de simulation sont les clés de ces enjeux, puisque certains des partenaires de ce projets (Dassault-Aviation, Dassault-Système, EDF, Renault,...) sont des sociétés représentant aujourd'hui les principaux acteurs qui ont recours à des simulations de systèmes complexes de très grandes tailles.
Parmi les résultats attendus, on peut citer :

- Une méthode d'adaptation de modèle d'ordre réduit permettant de choisir entre des simulations à qualité contrôlée ou des simulations à durée limitée,
- Un démonstrateur illustrant la méthode de recalage de paramètres,
- Des méthodes d'enrichissement de surfaces de réponses, dont des méthodes d'interpolation ou d'adaptation de bases POD.

Nous nous sommes principalement intéressés aux deux premiers points, que nous présenterons tout au long ce mémoire. Le troisième point est une perspective pour l'utilisation des résultats fournis dans ce mémoire.

En mécanique non linéaire des matériaux et des structures, ces modèles sont des modèles paramétrés. En effet, l'état mécanique d'un système peut être vu comme une fonction dépendant non seulement du temps et des variables

de l'espace, mais aussi de paramètres tels que des coefficients matériaux, des conditions de chargement, aux limites ou aux conditions initiales du modèle. Nous considèrerons la dépendance vis à vis de ces paramètres de deux manières : séquentielle (répétition de simulations similaires pour différentes valeurs de paramètres) ou multidimensionnelle (calcul simultané de différents modèles).
Le principal objectif de cette thèse est de proposer une méthode de simulation bien adaptée aux problèmes d'optimisation traités en milieu industriel ou en laboratoire.
Il y a deux types d'approches en optimisation : la résolution de problèmes inverses (correspondant à la manière séquentielle) et la construction de surfaces de réponses (manière multidimensionnelle).
Ces deux approches sont traitées à l'aide d'une méthode de réduction adaptative de modèles.
En effet, le coût de la résolution numérique des problèmes non linéaires peut rapidement devenir prohibitif, plus précisément pour les problèmes industriels complexes introduisant des modèles Éléments Finis de plusieurs centaines de milliers de degrés de liberté. Ainsi, si on a besoin de multiplier les simulations (dans le cas de problèmes d'optimisation par exemple), l'utilisation des calculs par Éléments Finis peut s'avérer assez lourde. Il y a donc une nécessité de réduire le nombre de degrés de liberté.
Les méthodes de réduction de modèles ("Reduced-Order Modeling" ou "ROM") ont été introduites pour remédier à ce problème. Elles visent à réduire le coût global pour l'obtention d'une solution en recherchant celle-ci dans l'espace d'une base plus physique plutôt que dans l'espace des fonctions de forme Éléments Finis.
Le principal objectif des méthodes de réduction de modèles est donc de remplacer l'espace des fonctions de forme Éléments Finis par un sous-espace de plus petite dimension, afin d'approximer la solution en utilisant les résultats de simulations précédentes.
L'utilisation des bases modales est très classique en mécanique. Mais une des stratégies les plus communes actuellement pour la construction de ces bases réduites est la POD (Proper Orthogonal Decomposition). Cette approche permet de réduire de manière significative le coût d'une résolution d'un système en introduisant des bases représentatives contenant très peu de fonctions de formes.

De plus, nous choisirons une technique adaptative, du fait que l'on doive gérer un grand nombre de simulations relatives aux problèmes d'optimisation. Il s'agit de la méthode APHR, introduite par David Ryckelynck en 2005, qui peut être vue comme une sorte de POD incrémentale. En effet,

contrairement aux approches classiques de type POD, elle propose de ne pas figer le modèle d'ordre réduit avant de traiter le problème d'optimisation. Une première version du ROM est choisie en début de procédure d'optimisation. Puis, ce modèle d'ordre réduit est actualisé au cours du processus d'optimisation.
En effet, la plupart des méthodes de réduction de modèle exploitent une base de données contenant des résultats de simulations coûteuses à obtenir et dont l'actualisation est également coûteuse en cas d'ajout de paramètres à étudier. L'approche choisie permet donc d'obtenir des modèles simplifiés capables de mieux capter les différentes sensibilités de la réponse du système aux variations des paramètres à optimiser.

L'originalité de l'approche développée consiste, pour la résolution de problèmes inverses, à tenir compte de la suite des prévisions déjà réalisées pour simplifier le modèle du système à optimiser à l'aide d'une méthode de réduction adaptative de modèles. Nous proposons de compléter cette méthode par une méthode d'atténuation des évènements récurrents apparaissant dans différentes prévisions. En effet, lors de l'adaptation du modèle de taille réduite au cours d'une nouvelle simulation, les évènements récurrents doivent être distingués des évènements particuliers relatifs à la simulation en cours. Du point de vue formel, un événement est le couple formé par un mode empirique et la variable d'état réduite associée qui est une fonction du temps, ou du temps et des paramètres. Lorsque la variable réduite d'un événement est significative pour chacune des prévisions considérées, l'événement est dit récurrent.
Un problème inverse peut être vu comme une suite de simulations qui diffèrent les unes des autres par une modifications de paramètres, jusqu'à convergence de l'algorithme d'optimisation. Nous allons montrer que, lors de l'adaptation du modèle de taille réduite au cours d'une nouvelle simulation, les évènements récurrents doivent être distingués des évènements particuliers relatifs à la simulation en cours. Cela est dû au fait que, comme nous le verrons dans le sous-chapitre 1.3.2 , réduire un modèle entraîne l'extraction d'une transformation des évènements les plus significatifs. Et ceux-ci seront d'autant plus significatifs qu'ils sont présents dans une transformation. Or, il existe plusieurs façons pour un évènement d'être significatif : soit il est très intense sur une durée limitée, soit il est moins intense sur une durée plus longue. La répétition de ces simulations (qui restent donc assez similaires) au cours du traitement de problèmes d'optimisation implique l'accroissement de l'importance de certains modes empiriques, et des évènements nécessaires à la conduite de l'optimisation peuvent devenir de moins en moins significatifs.

Concernant la construction de surfaces de réponses, l'originalité de l'approche consiste à développer une méthode numérique de représentation à variables séparées pour la représentation de problèmes paramétriques. La réduction de modèle permet de séparer les variables d'espaces et les variables du temps. Nous chercherons à séparer les variables d'espace et les variables de paramètres. Il s'agit en fait de passer d'un système multidimensionnel à des simulations simultanées. On écrit la réponse sur un système particulier : une somme de systèmes découplés qui correspond à une simulation simultanée de différents problèmes. Pour un incrément de temps donné, plusieurs valeurs de paramètres seront traitées de manière simultanée sur l'ensemble du système.

Nous voulons prouver que la méthode de réduction adaptative de modèle APHR est capable d'estimer efficacement les variables mécaniques. Pour cela, le résidu des équations d'équilibre est introduit dans la formulation des équations de ce mémoire. En effet, le but d'une simulation numérique est de prévoir l'état mécanique relatif à un résidu assez petit, ou aussi petit que possible.

Ce mémoire sera organisé de la manière suivante :

- Dans le chapitre 2, nous présenterons un état de l'art sur les méthodes de construction d'approximation par séparation de variables, ainsi que sur les méthodes de réduction de modèles, et leur application en science des matériaux. L'utilisation de ces différentes méthodes d'approximation pour l'optimisation fera également l'objet d'une section de ce chapitre.
- Le chapitre 3 sera consacré à la formulation de la méthode adaptative de réduction de modèle APHR en mécanique.
- Dans le chapitre 4, nous nous intéresserons à l'utilisation de la méthode APR pour résoudre des suites de simulations. Nous étudierons les évènements récurrents, prouverons leur existence, et développerons une solution afin de les atténuer. Cette solution nous permettra de traiter des problèmes inverses en utilisant la méthode APR sans prendre le risque de détériorer la qualité des calculs.
- Le chapitre 5 traitera des simulations simultanées pour construire des surfaces de réponses. Pour cela, la méthode APHR sera étendue au traitement de ce type de problème.
- Le chapitre 6 fera office de résumé de ce travail de thèse, suivi des

conclusions et des perspectives des travaux à venir.

Tout au long de ce mémoire de thèse, nous prendrons un exemple numérique qui nous servira de fil conducteur. Il s'agit d'un problème mécanique classique : un problème purement académique d'une éprouvette en traction simple, traité en 3D. Ce problème est détaillé en section 2.5.

Evidemment, nous traiterons également d'autres types de problèmes, avec différents matériaux, car l'approche développée est générique. Le choix du matériau n'influe pas sur la méthode. Il s'agit toutefois de problèmes à lois de comportement réalistes traités au sein du Centre des Matériaux.

Chapitre 1

État de l'art

1.1 Méthodes de construction d'approximation ayant une représentation à variables séparées

1.1.1 Contexte et notations

Considérons un modèle paramétré et notons $\{p\}$ le vecteur colonne contenant les paramètres du modèle.
Considérons une suite de problèmes relatifs à une suite de valeurs de paramètres $(\{p\}_\alpha)_{\alpha=1,...,n_p}$.

Soit Ω le domaine occupé par le milieu continu à l'instant t. Le système non linéaire ainsi formulé est analysé sur l'intervalle $]0,\ T]$.
Le champ inconnu au temps t est défini sur Ω et est noté $\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha)$.
$\underline{\boldsymbol{X}}$ représente la position initiale d'un point dans Ω.

Puisque l'approche que nous proposons de développer peut être appliquée à d'autres problèmes différentiels, nous nous placerons dans un cas général. La forme générale de ces problèmes est notée $\{\mathcal{L}\}$ et est donnée par :

Trouver la fonction inconnue $\underline{\boldsymbol{u}}\,(.,.;\{p\}_\alpha)$ *définie sur* $\Omega\times]0,T]$ *tel que :*

$$\mathcal{L}:\left\{\begin{array}{l}\mathcal{L}\left(\underline{\boldsymbol{X}}\,, t, \{p\}_\alpha\,, \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha), \left(\dfrac{\partial^\beta \underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^{\,\beta}}(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha)\right), \dfrac{\partial u}{\partial t}(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha)\right) = 0, \\ (\beta = 1, ..., \zeta) \\ CI + CL \quad \textit{(Conditions initiales et aux limites)}\end{array}\right. \tag{1.1}$$

où $\mathcal{L}$ est un opérateur différentiel représentant l'équation aux dérivées partielles qui relie le champ inconnu $\underline{\boldsymbol{u}}$ à ses dérivées.
On suppose que cette forme différentielle ne fait pas intervenir de dérivées partielles par rapport aux composantes du vecteur des paramètres $\{p\}$. En conséquence, les paramètres n'induisent pas de condition de régularité sur les champs recherchés.
Soit $\partial\Omega$ la frontière du domaine Ω, cette frontière se décompose de manière classique telle que : $\partial\Omega = \partial_U\Omega \cup \partial_f\Omega$.
Sur la partie $\partial_f\Omega$, les efforts extérieurs agissant sur le domaine Ω sont donnés. La densité surfacique de ces efforts est notée $\underline{\boldsymbol{f}}(.,t;.)$. Ainsi, sur la partie $\partial_f\Omega$, on impose un champ d'effort donné $\underline{\boldsymbol{f}}(.,t;.)$ dépendant du temps t.
Sur la partie $\partial_U\Omega$, on impose les conditions aux bords de type Dirichlet (1.2) :

$$\underline{\boldsymbol{u}}(.,t;\{p\}_\alpha) = \underline{\boldsymbol{u}}_c(.,t;\{p\}_\alpha)\forall t. \tag{1.2}$$

où $\underline{\boldsymbol{u}}_c$ est donné.
Le champ inconnu $\underline{\boldsymbol{u}}$ appartient à un espace de fonctions affines $\mathcal{U}$ défini par :

$$\mathcal{U} = \left\{\underline{\boldsymbol{u}}(.,t;\{p\}_\alpha) \in H^1(\Omega) \mid \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}},t;\{p\}_\alpha) = \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}},t) \quad \forall\, \underline{\boldsymbol{X}} \in \partial_U\Omega\right\}. \tag{1.3}$$

L'espace vectoriel associé à $\mathcal{U}$ est noté $\mathcal{V}$. On suppose qu'il ne dépend d'aucun paramètre. Nous n'aborderons pas dans ce mémoire le cas de l'optimisation topologique.

$$\mathcal{V} = \left\{\underline{\boldsymbol{u}}(.,t;\{p\}_\alpha) \in H^1(\Omega) \mid \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}},t;\{p\}_\alpha) = 0\right\} \tag{1.4}$$

Les méthodes classiques de discrétisation (Éléments Finis, Volumes Finis, ...) des équations aux dérivées partielles ou des équations différentielles ordinaires issues de la modélisation des processus physiques permettent d'obtenir des solutions numériques approchées. En général, on applique des méthodes à résidus pondérés pour chercher des approximations des solutions pour des problèmes différentiels. Ces méthodes à résidus pondérés se basent sur la recherche de la meilleure approximation possible de la solution du problème (1.1) qui réduit au minimum le résidu pondéré par un champ de pondération (ou champ test) $\underline{\boldsymbol{u}}^*$, et cela, d'une manière intégrale sur tout le domaine Ω. Il s'agit en fait d'annuler le résidu pondéré r défini par

$$r(\underline{\boldsymbol{u}},\underline{\boldsymbol{u}}^*,t,\{p\}_\alpha) = \int_\Omega \mathcal{L}\left(\underline{\boldsymbol{X}},t,\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}},t;\{p\}_\alpha),\frac{\partial^\beta \underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^\beta},\frac{\partial \underline{\boldsymbol{u}}}{\partial t}\right)\underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}})d\Omega \tag{1.5}$$

La méthode de Galerkin est un cas particulier des méthodes à résidus pondérés où les champs test sont choisis dans le même espace que les champs

de base utilisés dans le développement de la solution approchée. Par conséquent, l'approximation par la méthode de Galerkin revient à résoudre alors :

Trouver $\underline{\boldsymbol{u}}\,(.,t)$ *dans l'espace fonctionnel adéquat* $\mathcal{U}$*, tel que :*

$$\int_{\Omega} \mathcal{L}\left(\underline{\boldsymbol{X}}\,, t, \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_{\alpha}), \frac{\partial^{\beta}\underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^{\,\beta}}, \frac{\partial \underline{\boldsymbol{u}}}{\partial t}\right) \underline{\boldsymbol{u}}^{\,*}(\underline{\boldsymbol{X}}\,) d\Omega \;=\; 0 \qquad \forall \underline{\boldsymbol{u}}^{\,*} \in \mathcal{V}, \forall t \in]0, T] \tag{1.6}$$

L'utilisation de la méthode des Éléments Finis [Zienkiewicz and Taylor, 2000], d'un schéma d'intégration temporelle (Euler implicite), et d'un algorithme de Newton-Raphson, nous permet de construire un modèle numérique du problème à réduire. On appellera ce modèle numérique le "modèle détaillé". Celui-ci introduit n_h degrés de libertés $(q_j(t; \{p\}_{\alpha}))_{j=1..n_h}$. Le modèle Éléments Finis de la formulation faible (1.6) donne une solution approchée $\underline{\boldsymbol{u}}_h$ dans l'espace des fonctions Éléments Finis $\mathcal{V}_h$ où N_j sont les fonctions de forme Éléments Finis relatives au maillage avec n_h degrés de libertés. Chaque composante q_j représente un déplacement nodal relié au champ des déplacements par les fonctions de forme $\underline{\boldsymbol{N}}_j$ tel que, pour une approche séquentielle :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_{\alpha}) = \sum_{j=1}^{j=n_h} \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,)\, q_j(t; \{p\}_{\alpha}). \tag{1.7}$$

où l'espace vectoriel $\mathcal{V}_h$ s'écrit :

$$\mathcal{V}_h = \text{span}\left\{\underline{\boldsymbol{N}}_1, ..., \underline{\boldsymbol{N}}_{n_h}\right\}. \tag{1.8}$$

Nous venons de définir un niveau d'approximation pour les déplacements relatif au modèle numérique Eléments Finis standard. Ce modèle est en fin de compte déduit d'un modèle continu (1.6) utilisant un sous espace de $\mathcal{V}$ (1.4) . Ce sous espace est noté $\mathcal{U}_h$ et est défini comme suit :

$$\begin{aligned} \mathcal{U}_h \;=\; \{\, \underline{\boldsymbol{u}} \in \mathcal{V}_h \,|\, \exists\, \{q\} \in \mathbb{R}^{n_h}\,, \\ \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_{\alpha}) = \sum_{j=1}^{j=n_h} \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,)\, \widehat{q}_j(t; \{p\}_{\alpha}) + \sum_{j=1}^{j=n_c} \widetilde{\underline{\boldsymbol{N}}}_j(\underline{\boldsymbol{X}}\,)\, g_j(t) \\ \forall \underline{\boldsymbol{X}} \in \Omega \quad \forall t \in]0, T] \,\}\,. \end{aligned} \tag{1.9}$$

Les fonctions de forme $(\widetilde{\underline{\boldsymbol{N}}}_j)_{j=1,...n_c}$ sont reliées aux noeuds appartenant à $\partial_U \Omega^0$. Les coefficients g_j représentent des déplacements donnés relatifs à $\underline{\boldsymbol{u}}_c$.

Le champ des déplacements qui correspond au relèvement des déplacements imposés est noté $\underline{\boldsymbol{u}}_{ch}$ tel que :

$$\underline{\boldsymbol{u}}_{ch}(\underline{\boldsymbol{X}}, t) = \sum_{j=1}^{j=n_c} \widetilde{\underline{\boldsymbol{N}}}_j(\underline{\boldsymbol{X}})\, g_j(t), \quad \forall \underline{\boldsymbol{X}} \in \Omega \tag{1.10}$$

Si l'on retranche la partie imposée $\underline{\boldsymbol{u}}_{ch}$ aux champs de déplacements $\underline{\boldsymbol{u}}$, nous obtenons une suite de déplacements continus $\underline{\boldsymbol{U}}_i \in \mathcal{V}_h$ pour $i = 1...m$. Les champs de déplacements $\underline{\boldsymbol{U}}_i$ sont relatifs à différentes valeurs de paramètres $\{p\}_\alpha$ à différents instants.
En remplaçant $\mathcal{U}$ par $\mathcal{U}_h$ et $\mathcal{V}$ par $\mathcal{V}_h$ dans la formulation Éléments Finis Galerkin du problème (1.6), nous obtenons le problème de référence (ou problème détaillé) à résoudre :

On cherche $\underline{\boldsymbol{u}} \in \mathcal{U}_h$ *tel que :*

$$\begin{aligned} \int_\Omega \mathcal{L}\left(\underline{\boldsymbol{X}}, t, \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha), \frac{\partial^\beta \underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^\beta}, \frac{\partial \underline{\boldsymbol{u}}}{\partial t}\right) \underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}}) d\Omega &= 0 \\ \forall \underline{\boldsymbol{u}}^* \in \mathcal{V}_h, \forall t \in]0, T] & \end{aligned} \tag{1.11}$$

Les méthodes de discrétisation classiques du problème (1.1), Éléments Finis notamment, se basent sur la méthode de Galerkin. Elle peuvent donc être considérées donc des méthodes de modèles réduits puisque l'on a approché un problème posé dans un espace de dimension infini par un autre posé dans un espace de dimension fini. Ici, l'ordre d'un modèle détaillé est n_h. Malgré cela, la dimension de l'espace de résolution des approximations de certains problèmes non linéaires complexes peut être très grande, ce qui la rend très coûteuse, voire impossible. En effet, certains problèmes aéronautiques réalistes [Spalart et al., 1997] peuvent nécessiter des centaines de milliards de points sur quelques millions de pas de temps.

Le développement de ces nombreux modèles Éléments Finis augmente donc le besoin d'avoir des modèles d'ordre moindre, créés par les méthodes de réduction de modèles. L'objectif de ces méthodes est de réduire le nombre d'équations nécessaires pour décrire le système physique que l'on veut étudier, afin de gagner en temps de calcul.
La formulation des équations réduites diffère de la formulation des équations détaillées que nous venons de voir, par le choix de l'espace de fonctions relatif aux variables d'états, comme nous le verrons en section 1.1.2.5

1.1.2 Les méthodes de réduction d'ordre de modèles

La plupart du temps, les variables d'état sont considérées comme une combinaison linéaire de champs connus. Dans le cas d'une approche "a posteriori", ces champs sont obtenus en résolvant les problèmes préliminaires détaillés. Dans le cadre d'une méthode de superposition modale, ces champs sont les modes propres définis par le problème de vibrations libres. Si les problèmes préliminaires fournissent des champs linéairement indépendants, alors les coefficients de la combinaison linéaire seront les variables d'états réduites du modèle d'ordre réduit. Dans le cas contraire, il faudra construire une base réduite du sous-espace engendré par les champs connus. La Décomposition Orthogonale aux valeurs Propres (POD) vise à créer une telle base à partir de champs dépendant du temps, et éventuellement de paramètres.

Nous allons suivre les démarches de Cordier et Bergmann [Cordier and Bergmann, 2006] pour présenter la méthode de Décomposition Orthogonale aux valeurs Propres (POD) [Karhunen, 1946] [Loeve, 1963], et présenter les approches initialement proposées par Lumley [Lumley, 1967] et Sirovich [Sirovich, 1987].

1.1.2.1 Problématique générale

La méthode POD consistera à définir à partir d'un ensemble de données $\{\underline{\boldsymbol{u}}\,(.,t;.)\}_t$ pour t parcourant un intervalle de temps discret ou continu, une base orthogonale dite "optimale" pour représenter $\underline{\boldsymbol{u}}$, solution d'un problème physique *i.e* :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) \simeq \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}\,)a_k(t;\{p\}_\alpha). \tag{1.12}$$

telle que cette approximation devienne exacte lorsque le nombre de termes s de la sommation devient infini. Cette décomposition n'est certainement pas unique, puisque nous avons vu (équation 1.7) que la méthode des Éléments Finis permet d'écrire $\underline{\boldsymbol{u}}$ de la sorte :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) = \sum_{j=1}^{j=n_h} \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,)\, q_j(t;\{p\}_\alpha).$$

Il est donc logique de chercher à construire la meilleure approximation possible pour une valeur de s donnée.

Le problème est alors de chercher : un nombre s, une base spatiale $\{\underline{\boldsymbol{\psi}}_k\}_{k=1}^{s}$

et des coefficient $\{a_k\}$ vérifiant (1.12). On cherchera la base spatiale orthonormée pour le produit scalaire canonique *i.e* :

$$(\underline{\boldsymbol{\psi}}_i, \underline{\boldsymbol{\psi}}_j)_\Omega = \delta_i^j, \quad \forall i, j = 1, \cdots, s \tag{1.13}$$

où le produit scalaire est défini par :

$$(\underline{\boldsymbol{u}}, \underline{\boldsymbol{v}})_\Omega = \int_\Omega \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}})\underline{\boldsymbol{v}}(\underline{\boldsymbol{X}})d\Omega \in \mathbb{R}, \underline{\boldsymbol{u}} \in \{L^2(\Omega)\}^d, \underline{\boldsymbol{v}} \in \{L^2(\Omega)\}^d \tag{1.14}$$

soit :

$$\int_\Omega \underline{\boldsymbol{\psi}}_i(\underline{\boldsymbol{X}})\underline{\boldsymbol{\psi}}_j(\underline{\boldsymbol{X}})\, d\Omega = \delta_i^j, \quad \forall i, j = 1, \cdots, s \tag{1.15}$$

où δ_i^j est le symbole de Kronecker. Si on arrive à obtenir la base spatiale $\{\underline{\boldsymbol{\psi}}_k\}_{k=1}^s$, les fonctions temporelles $\{a_k\}_{k=1}^s$ sont déduites de la relation (1.12) grâce à l'orthogonalité des fonctions de base spatiales $\{\underline{\boldsymbol{\psi}}_k\}_{k=1}^s$ et au produit scalaire défini en (1.15) telles que :

$$a_k(t; \{p\}_\alpha) = \int_\Omega \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha)\underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}})d\Omega \quad \forall k = 1, \cdots, s. \tag{1.16}$$

On peut utiliser le produit scalaire défini en (1.14) pour écrire a_k sous la forme :

$$a_k(t; \{p\}_\alpha) = \left(\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha), \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}})\right)_\Omega \tag{1.17}$$

Supposons connues les valeurs de $\underline{\boldsymbol{u}}(., .; \{p\}_\alpha)$ en N_x localisations spatiales : $X_1, X_2, ..., X_{N_x}$ et N_t instants différents : $t_1, t_2, ..., t_{N_t}$. Généralement, elles sont obtenues grâce au modèle de référence (par une méthode des Éléments Finis).
Il est évident que la meilleure approximation possible pour (1.12) s'obtient en minimisant l'écart entre la solution exacte connue $\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha)$ et sa valeur approchée $\sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}})a_k(t; \{p\}_\alpha)$.

En utilisant les notations de (1.17), il s'agit donc de déterminer la base orthonormée $\{\underline{\boldsymbol{\psi}}_k\}_{k=1}^{s}$ solution du problème de minimisation :

$$min \quad \sum_{i=1}^{N_t} \; ||\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t_i; \{p\}_\alpha) - \sum_{k=1}^{s} \Big(\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t_i; \{p\}_\alpha), \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}\,)\Big)_\Omega \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}\,)||^2. \tag{1.18}$$

avec $||\underline{\boldsymbol{v}}\,||^2 = (\underline{\boldsymbol{v}}\,, \underline{\boldsymbol{v}}\,)_\Omega$

L'ensemble des réalisations connues (également appelées "snapshots")

$$\mathcal{S} = \{\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t_i; \{p\}_\alpha)\}_{i=1..N_t}$$

peut être rangé dans la matrice $[Q_\alpha]$ des snapshots suivante :

$$[Q_\alpha] = \begin{bmatrix} \mathbf{u}(\mathbf{x}_1, t_1; \{p\}_\alpha) & \cdots & \mathbf{u}(\mathbf{x}_1, t_{N_t}; \{p\}_\alpha) \\ \vdots & \vdots & \vdots \\ \mathbf{u}(\mathbf{x}_{N_x}, t_1; \{p\}_\alpha) & \cdots & \mathbf{u}(\mathbf{x}_{N_x}, t_{N_t}; \{p\}_\alpha) \end{bmatrix}. \tag{1.19}$$

Lorsque les champs connus sont des solutions d'un problème aux Éléments Finis, on a :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t_i; \{p\}_\alpha) = \sum_{j=1}^{n_h} \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,) Q_{\alpha,ji} \tag{1.20}$$

Soit :

$$M_{ij} = \sum_{p=1}^{n_h}\sum_{q=1}^{n_h} \left(Q_{\alpha,qi} \int_\Omega \underline{\boldsymbol{N}}_p(\underline{\boldsymbol{X}}\,) \underline{\boldsymbol{N}}_q(\underline{\boldsymbol{X}}\,) d\Omega Q_{\alpha,pj} \right) \tag{1.21}$$

$$= \left([Q]_\alpha^T [m] [Q]_\alpha \right)_{ij} \tag{1.22}$$

où $[m]$ est la matrice des composantes :

$$m_{ij} = \int_\Omega \underline{\boldsymbol{N}}_i(\underline{\boldsymbol{X}}\,) \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,) d\Omega \tag{1.23}$$

En pratique, nous ne calculons pas $[m]$. Celle-ci étant une matrice symétrique définie positive, on peut lui appliquer la décomposition de Cholesky et trouver ainsi une matrice $[L]$ triangulaire inférieure telle que :

$$[m] = [L] [L]^T \tag{1.24}$$

On obtient alors :

$$[M] = [Q]_\alpha^T [L] [L]^T [Q]_\alpha \tag{1.25}$$

Alors la solution du problème de minimisation (1.18) est donnée par la Décomposition aux Valeurs Singulières (SVD) tronquée à l'ordre s de la matrice $[L]^T [Q_\alpha]$.

Nous cherchons les vecteurs propres de $[Q]_\alpha^T [Q]_\alpha$ pour obtenir les composantes b_{ik}, et écrire :

$$\underline{\boldsymbol{\psi}}_k = \sum_{i=1}^{N_t} \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t_i; \{p\}_\alpha) b_{ik} \tag{1.26}$$

$$= \sum_{j=1}^{n_h} \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}) A_{jk} \tag{1.27}$$

où

$$A_{jk} = \sum_{i=1}^{m} Q_{\alpha,ji} b_{ik} \tag{1.28}$$

sont les composantes de la matrice $[A]$, la matrice de réduction de base.

Cette matrice définie, on peut écrire :

$$\{q\}(t) = [A] \cdot \{a\}(t) \tag{1.29}$$

1.1.2.2 Décomposition aux valeurs singulières (SVD)

Soit $[Q]$ une matrice à valeurs réelles rectangulaire de dimension $N_x \times N_t$ et de rang r. La Décomposition aux Valeurs Singulières est la factorisation :

$$[Q] = [U] [\Sigma] [V]^T \tag{1.30}$$

avec $[U]$ et $[V]$, deux matrices orthogonales, respectivement de taille $N_x \times N_x$ et $N_t \times N_t$ et

$$[\Sigma] = \begin{pmatrix} [\Sigma]_r & 0 \\ 0 & 0 \end{pmatrix} \tag{1.31}$$

où $[\Sigma]_r$ est la matrice carrée diagonale de taille r contenant les valeurs singulières de $[Q]$, notées $(\sigma_i)_{i=1\ldots r}$ rangées par ordre décroissant, tel que $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

1.1.2.3 Lien entre SVD et problème aux valeurs propres

Nous pouvons remarquer qu'en multipliant les deux membres de l'équation (1.30) par $[Q]^T$ à gauche (resp. à droite), et en se rappelant que $[U]$ (resp. $[V]$) est une matrice orthogonale, *i.e* $[U]^T [U] = [\mathbb{I}]_{N_x}$, (resp. $[V] [V]^T = [\mathbb{I}]_{N_t}$), on obtient très facilement :

$$[Q]^T [Q] = [V]^T [\Sigma]^2 [V] \tag{1.32}$$

$$(\text{resp.} \quad [Q] [Q]^T = [U] [\Sigma]^2 [U]^T \quad) \tag{1.33}$$

On peut poser $[\Lambda] = [\Sigma]^2$, alors les valeurs propres λ_i de $[\Lambda]$ vérifient $\sigma_i = \sqrt{\lambda_i}$.

Ainsi, la décomposition aux valeurs singulières devient un problème aux valeurs propres lorsque l'on applique la SVD aux matrices de covariance $[Q]^T [Q]$ (resp. $[Q] [Q]^T$).
$([V], [\Lambda])$ représente donc la décomposition aux valeurs propres de la matrice $[Q]^T [Q]$ de taille N_t, (resp. $([U], [\Lambda])$ représente la décomposition aux valeurs propres de la matrice $[Q] [Q]^T$ de taille N_x).
Ainsi, lorsque $N_t << N_x$, la résolution du problème aux valeurs propres lié à la matrice $[Q]^T [Q]$ est bien moins coûteuse que celle concernant la matrice $[Q] [Q]^T$.

1.1.2.4 La méthode Proper Orthogonal Decomposition (POD)

Il existe donc deux approches différentes pour la POD : la méthode classique introduite par Lumley [Lumley, 1967] (liée à $[Q] [Q]^T$) et la méthode des snapshots imaginée par Sirovich [Sirovich, 1987] (relative à $[Q]^T [Q]$).
Ces méthodes POD (processus déterministe) proviennent toutes les deux de l'expansion de Karhunen-Loève (processus stochastique) [Karhunen, 1946] [Loeve, 1963] développée pour les analyses statistiques.
L'expansion de Karhunen-Loève est largement utilisée dans les méthodes de réduction de modèle a posteriori de problèmes non linéaires dépendant du temps. Les premiers travaux concernant cette approche portaient sur la prévision météorologique [Lorenz, 1956] grâce à une analyse statistique de données expérimentales (pression et températures).
La méthode POD classique a été introduite par Lumley [Lumley, 1967] en mécanique non linéaire des fluides, non pas pour proposer un solveur, mais comme un moyen d'interpréter des écoulements turbulents. Il s'agissait d'extraire les structures cohérentes d'un champ de vitesse d'un écoulement, *i.e.*

trouver les fonctions déterministes qui sont les mieux corrélées en moyenne aux observations. Il a notamment mis en avant que ces fonctions propres $\{\psi_k\}$ sont représentatives d'un point de vue énergétique. Ainsi, considérer uniquement les premières fonctions $\{\psi_k\}$ permet de rendre compte des principaux phénomènes énergétiques mis en jeu.

Mais l'approche la plus exploitée pour les problèmes non linéaires semble être la Snapshot POD proposée en [Sirovich, 1987] car elle a permis de réduire la taille du problème aux valeurs propres (voir 1.32) à résoudre pour construire les vecteurs POD, par rapport à celui fournit par l'expansion classique de Karhunen-Loève. Elle a donc eu une contribution importante dans le développement des bases POD pour la réduction de problèmes non-linéaires dépendant du temps.
Cette méthode a connu un véritable essor depuis la fin des années 90 dans différents domaines et c'est celle-ci que nous utiliserons.

Quelle que soit l'approche utilisée, la méthode POD permet de déterminer la famille de fonctions orthogonales $\{\underline{\boldsymbol{\psi}}_k\}_{k=1\ldots s}$, solution du problème de minimisation (1.18) par décomposition aux valeurs propres de $[Q]\,[Q]^T$ ou $[Q]^T\,[Q]$ tronqué à l'ordre s.
Il s'agit donc de rechercher les fonctions ψ les mieux corrélées en moyenne, aux snapshots $\mathcal{S}$, c'est-à-dire celles qui possèdent, au sens des moindres carrés, la plus grande projection sur les observations, *i.e* qui maximise la quantité $|(\underline{\boldsymbol{u}}\,,\underline{\boldsymbol{\psi}}\,)|$.
Pour ne pas avoir à distinguer les différentes approches de la méthode POD, nous allons traiter de manière générale la méthode de recherche de ces fonctions $\underline{\boldsymbol{\psi}}$.
Il faut donc choisir un espace des fonctions $\underline{\boldsymbol{\psi}}$ pour lequel le produit scalaire existe, on impose donc à $\underline{\boldsymbol{\psi}}$ d'appartenir à l'espace des fonctions carré intégrable sur Ω, soit $L^2(\Omega)$.

Grâce à ce produit scalaire, la décomposition de Karhunen-Loève est définie comme un problème d'optimisation sous contrainte où un champ $\underline{\boldsymbol{\psi}}$ doit maximiser sa projection $\lambda(\underline{\boldsymbol{\psi}}\,)$ sur tous les champs engendrés par les variables d'état $\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t_i;\{p\}_\alpha)$ (FIG. 1.1).

$\lambda(\underline{\boldsymbol{\psi}}\,)$ est donc défini tel que :

$$\lambda(\underline{\boldsymbol{\psi}}\,) = \frac{\displaystyle\int_0^T (\underline{\boldsymbol{u}}\,,\underline{\boldsymbol{\psi}}\,)_\Omega^2 dt}{(\underline{\boldsymbol{\psi}}\,,\underline{\boldsymbol{\psi}}\,)_\Omega} \tag{1.34}$$

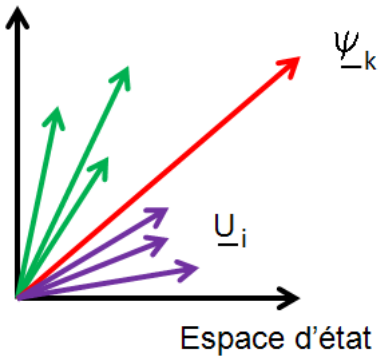


FIG. 1.1 – Maximiser la projection de $\underline{\boldsymbol{\psi}}_k$ sur l'ensemble des réponses $\underline{\boldsymbol{u}}$

Il s'agit donc de résoudre le problème de maximisation suivant :

$$\max_{\underline{\boldsymbol{\psi}} \in L^2(\Omega)} \lambda(\underline{\boldsymbol{\psi}}). \tag{1.35}$$

Nous allons montrer que ce problème d'optimisation conduit à un problème aux valeurs propres (équation 1.40) dont les solutions sont les vecteurs propres empiriques $\underline{\boldsymbol{\psi}}_k$.
En effet, si l'on pose l'opérateur $\mathcal{R}_{co} : L^2(\Omega) \rightarrow L^2(\Omega)$ défini pour tout $\underline{\boldsymbol{\psi}} \in L^2(\Omega)$ par

$$\mathcal{R}_{co}\underline{\boldsymbol{\psi}}(\underline{\boldsymbol{X}}) = (\mathcal{R}_{co}(\underline{\boldsymbol{X}}, .), \underline{\boldsymbol{\psi}}) = \int_\Omega \underset{\sim}{\boldsymbol{R}}_{co}(\underline{\boldsymbol{X}}, \underline{\boldsymbol{X}}')\underline{\boldsymbol{\psi}}(\underline{\boldsymbol{X}}')d\Omega. \tag{1.36}$$

où $\underset{\sim}{\boldsymbol{R}}_{co}(\underline{\boldsymbol{X}}, \underline{\boldsymbol{X}}')$ est le tenseur des corrélations spatio-temporelles en deux points :

$$\underset{\sim}{\boldsymbol{R}}_{co}(\underline{\boldsymbol{X}}, \underline{\boldsymbol{X}}') = \int_0^T \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha) \otimes \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}', t; \{p\}_\alpha)dt \tag{1.37}$$

Dans ce cas, on obtient [Cordier and Bergmann, 2006] :

$$\left(\mathcal{R}_{co}\underline{\boldsymbol{\psi}}, \underline{\boldsymbol{\psi}}\right) = \int_0^T \underline{\boldsymbol{u}}, \underline{\boldsymbol{\psi}})^2 dt \geq 0 \tag{1.38}$$

$$\text{et } \left(\mathcal{R}_{co}\underline{\boldsymbol{\psi}}, \underline{\boldsymbol{\phi}}\right) = \left(\mathcal{R}_{co}\underline{\boldsymbol{\phi}}, \underline{\boldsymbol{\psi}}\right), \quad \forall \underline{\boldsymbol{\psi}}, \underline{\boldsymbol{\phi}} \in L^2(\Omega) \tag{1.39}$$

L'opérateur $\mathcal{R}_{co}$ est donc un opérateur linéaire, auto-adjoint et positif sur $L^2(\Omega)$, alors le problème (1.35) est équivalent au problème aux valeurs propres :

$$\mathcal{R}_{co}\underline{\boldsymbol{\psi}} = \lambda\underline{\boldsymbol{\psi}}. \tag{1.40}$$

Ce problème admet une infinité dénombrable de solutions $\{\lambda_k, \underline{\boldsymbol{\psi}}_k\}_{k=1}^{\infty}$. Ces solutions sont telles que :

$$\int_{\Omega} \underset{\sim}{\boldsymbol{R}}_{co}(\underline{\boldsymbol{X}}, \underline{\boldsymbol{X}}')\underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}')d\Omega = \lambda_k \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}). \tag{1.41}$$

Chaque fonction propre est déterminée comme solution du problème d'optimisation sous la contrainte $||\underline{\boldsymbol{\psi}}_k|| = 1$. Par conséquent, la solution du problème de maximisation (1.35), qui est le premier vecteur de la décomposition POD, est le vecteur propre correspondant à la plus grande valeur propre.
L'optimalité de la POD d'un point de vue énergétique a été étudiée en [Holmes et al., 1997]. Il s'en suit un résultat optimal pour la POD : quelquesoit s, la projection sur le sous-espace engendré par les s premiers modes d'une décomposition orthogonale aux valeurs propres contient en moyenne la plus grande quantité d'énergie cinétique possible.
Ainsi, la base POD étant optimale au sens de l'énergie, seul un très petit nombre de modes POD est suffisant pour obtenir une bonne représentation de $\underline{\boldsymbol{u}}$, et donc décrire correctement l'évolution d'état du système.

L'opérateur $\mathcal{R}_{co}$ étant auto-adjoint et positif, toutes ses valeurs propres sont réelles et positives. On peut donc les renuméroter, et les classer de manière décroissante telle que :

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq \cdots \geq 0. \tag{1.42}$$

À chaque $\underline{\boldsymbol{\psi}}_k$ correspond une valeur propre $\lambda_k = \lambda(\psi_k)$. Ces valeurs propres servent à ordonner les vecteurs propres empiriques $\underline{\boldsymbol{\psi}}_k$ selon la valeur des projections λ_k associées, une partie de ces vecteurs ayant une contribution négligeable à l'approximation (1.12). Dans la pratique, on sélectionne les s premières colonnes de $[b]$ qui forment la base réduite POD, tel que :

$$\sum_{k=1}^{k=s} \lambda_k = (1 - \varepsilon_{POD}) \sum_{k=1}^{\infty} \lambda_k \tag{1.43}$$

où ε_{POD} est un paramètre positif de l'approche snapshot POD permettant de définir le seuil de la troncature. Sa valeur par défaut est fixée à $\varepsilon_{POD} = 10^{-8}$.

L'idée de la snapshot POD est d'écrire les fonctions propres $\underline{\boldsymbol{\psi}}(\underline{\boldsymbol{X}})$ de la manière suivante :

$$\underline{\boldsymbol{\psi}}\left(\underline{\boldsymbol{X}}\right) = \sum_{i=1}^{N_t} b(t_i)\underline{\boldsymbol{u}}\left(\underline{\boldsymbol{X}}, t_i; \{p\}_\alpha\right), \tag{1.44}$$

on recherche alors les coefficients $b(t_k)_{(k=1,\ldots,N_t)}$ tels que les fonctions propres $\underline{\boldsymbol{\psi}}$ soient solutions de l'équation (1.41). En remplaçant $\int_0^T f dt$ par $\sum_{i=1}^{N_t} \delta t f(t_i)$, et après quelques développements, le problème aux valeurs propres à résoudre est le suivant :

$$[C]\{b\}_k = \lambda \{b\}_k \tag{1.45}$$

avec

$$\{b\} = \left\{ \begin{array}{c} b(t_1) \\ \vdots \\ b(t_i) \\ \vdots \\ b(t_{N_t}) \end{array} \right\} \tag{1.46}$$

et $[C]$ la matrice des corrélations temporelles définie par :

$$C_{ij} = \int_\Omega \underline{\boldsymbol{u}}\left(\underline{\boldsymbol{X}}, t_i; \{p\}_\alpha\right) \cdot \underline{\boldsymbol{u}}\left(\underline{\boldsymbol{X}}, t_j; \{p\}_\alpha\right) d\Omega \tag{1.47}$$

Ce problème, comme nous l'avons déjà précisé, est de taille inférieure à celui obtenu par la méthode POD classique.
L'inconvénient de la méthode snapshot POD est l'absence d'un critère qui permet de choisir des snapshots et des simulations préléminaires (valeurs de paramètres) satisfaisants pour créer les snapshots. En effet, la qualité d'une approximation POD dépend de l'extension du sous-espace engendré par les snapshots. Si l'approximation POD n'est pas assez précise, de nouveaux snapshots doivent être ajoutés à la suite de déplacements connus pour étendre la base POD.

1.1.2.5 Formulation de l'approximation réduite POD-Galerkin

Le but de cette section est de présenter le principe général de la méthode des bases réduites.
Le principal objectif des méthodes de réduction de modèles est de remplacer

$\mathcal{U}_h$ et $\mathcal{V}_h$ par des sous-espaces de plus petite dimension, afin d'approximer $\underline{\boldsymbol{u}}\,(.,t;\{p\}_\alpha)$ en utilisant les résultats de simulations précédentes.
Ces méthodes permettent de conserver l'ensemble des paramètres du modèle détaillé dans le modèle d'ordre réduit. Lorsqu'un sous-espace satisfaisant a été trouvé, il est nécessaire de reformuler les conditions d'équilibre au sens faible qui lui sont associées.
Soit donc $(\underline{\boldsymbol{U}}_i)_{i=1...m}$ une suite de solutions en déplacements obtenue par des simulations précédentes, telle que $\underline{\boldsymbol{U}}_i \in \mathcal{V}_h$. Le but de la méthode snapshot POD est d'utiliser cette suite comme "snapshots" (images instantanées) pour construire un modèle d'ordre réduit (ROM) : une base du sous-espace engendré par ces champs de déplacements, soit une base réduite de $\mathcal{V}_h$.

Soit $(\psi_k)_{k=1...s}$ les vecteurs de la base réduite POD (dans la pratique et afin d'avoir des gains de performances, on a $s << n_h$). Soit $\mathcal{V}_{POD}$ l'espace vectoriel relatif au modèle d'ordre réduit :

$$\mathcal{V}_{POD} = \text{span}\left\{\underline{\boldsymbol{\psi}}_1, ..., \underline{\boldsymbol{\psi}}_s\right\}. \tag{1.48}$$

L'approximation de la solution du modèle numérique précité sera d'autant plus précise sous réserve que les modifications du modèle restent dans la limite de validité de la base POD pré-calculée. Cette approximation peut s'écrire sous la forme :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}\,)\,a_k(t;\{p\}_\alpha) + \underline{\boldsymbol{u}}_{ch}(\underline{\boldsymbol{X}}\,,t), \quad \forall \underline{\boldsymbol{X}} \in \Omega \quad \forall t \in]0,T] \tag{1.49}$$

Remarque :
Ces vecteurs de la base réduite POD sont également appelés "fonctions de formes" dans le cadre de la théorie de l'approximation, ou "modes empiriques" [Lorenz, 1956]. Nous pouvons trouver en section 2.5 un exemple des quatre premiers modes empiriques obtenus après une première simulation en modèle d'ordre réduit sur le modèle "fil conducteur".

Les facteurs $(a_k)_{k=1...s}$ sont les variables d'état réduites du modèle d'ordre réduit. Ce sont des variables globales. L'espace affine relatif à l'approximation de la base réduite (1.49) est noté $\mathcal{U}_{POD}$:

$$\mathcal{U}_{POD} = \{\, \underline{\boldsymbol{u}} \in \mathcal{U} \mid \quad \underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}_{ch} \in \mathcal{V}_{POD} \,\} \tag{1.50}$$

Nous venons de définir un autre niveau d'approximation pour les déplacements relatifs au modèle d'ordre réduit POD. Ce dernier modèle est déduit du modèle continu (1.6) utilisant un sous espace de $\mathcal{V}$ (1.4) noté $\mathcal{U}_{POD}$, et défini comme suit :

$$\begin{aligned}\mathcal{U}_{POD} \;=\; & \{\, \underline{\boldsymbol{u}} \in \mathcal{V} \,|\exists\, \{a\} \in \mathbb{R}^s \,, \\ & \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha) = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}\,)\, a_k(t; \{p\}_\alpha) + \underline{\boldsymbol{u}}_{ch}(\underline{\boldsymbol{X}}\,, t) \\ & \forall \underline{\boldsymbol{X}} \in \Omega \quad \forall t \in]0, T] \,\}\,. \end{aligned} \tag{1.51}$$

Il est évident que $\mathcal{V}_{POD}$ est à son tour aussi un sous espace de $\mathcal{V}_h$. Les équations du modèle d'ordre réduit peuvent être obtenues simplement par une formulation de Galerkin en remplaçant $\mathcal{U}$ par $\mathcal{U}_{POD}$ et $\mathcal{V}$ par $\mathcal{V}_{POD}$ dans la formulation continue (1.6). Le problème réduit à résoudre est :

Trouver $\underline{\boldsymbol{u}} \in \mathcal{U}_{POD}$ *tel que :*

$$\begin{aligned}\int_\Omega \mathcal{L}\left(\underline{\boldsymbol{X}}\,, t, \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t), \frac{\partial^\beta \underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^{\,\beta}}, \frac{\partial \underline{\boldsymbol{u}}}{\partial t}\right) \underline{\boldsymbol{u}}^{\,*}(\underline{\boldsymbol{X}}\,) d\Omega \;&=\; 0, \\ \forall \underline{\boldsymbol{u}}^{\,*} \in \mathcal{V}_{POD}, \forall t \in]0, T], \quad \forall \underline{\boldsymbol{X}} \in \partial_U \Omega & \end{aligned} \tag{1.52}$$

Le résidu est celui défini en (1.5), où l'on remplace $\underline{\boldsymbol{u}}^{\,*}$ par $\underline{\boldsymbol{N}}_j$:

$$\mathcal{R}_j = \int_\Omega \mathcal{L}\left(\underline{\boldsymbol{X}}\,, t, \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha), \frac{\partial^\beta \underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^{\,\beta}}, \frac{\partial \underline{\boldsymbol{u}}}{\partial t}\right) \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,) d\Omega \tag{1.53}$$

Ce qui est équivalent à formuler l'équation d'équilibre (1.11) sous la forme :

$$\{\mathcal{R}\}\,(\{q\}\,, t; \{p\}_\alpha) = 0 \tag{1.54}$$

avec

$$\{\mathcal{R}\} = \left\{ \begin{array}{c} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_j \\ \vdots \\ \mathcal{R}_{n_h} \end{array} \right\} \tag{1.55}$$

Une autre formulation du résidu s'obtient en choisissant $\underline{\boldsymbol{u}}^{\,*} \in \mathcal{V}_{POD}$, soit :

$$\underline{\boldsymbol{u}}^{\,*} = \sum_{k=1}^{s} \sum_{j=1}^{n_h} \underline{\boldsymbol{N}}_j A_{jk} a_k^* \tag{1.56}$$

Ainsi, la formulation POD-Galerkin (1.52) peut aussi s'écrire sous forme matricielle condensée de la manière suivante :

$$[A]^T \cdot \{\mathcal{R}\}([A] \cdot \{a\}, t; \{p\}_\alpha) = 0 \tag{1.57}$$

La qualité du modèle d'ordre réduit peut être vérifiée grâce au résidu $\{\mathcal{R}\}$ de la condition d'équilibre Éléments Finis défini en (1.5). Une erreur relative peut être ainsi définie :

$$\eta = \max_{t\in[0,T]} \frac{\|\{\mathcal{R}\}([A] \cdot \{a\}, t; \{p\}_\alpha)\|}{\|\{\mathcal{R}\}(0, t; \{p\}_\alpha)\|} \tag{1.58}$$

Si $\eta = 0$, alors on connaît $\underline{\boldsymbol{u}} \in \mathcal{V}_{POD} \subset \mathcal{U}$ avec $\{\mathcal{R}\} = 0$ donc on a une solution du problème détaillé.
L'estimateur d'erreur η nous fournit donc le critère sur la qualité de l'approximation réduite qui peut être évaluée par le résidu des équations d'équilibre $\|\{\mathcal{R}\}([A] \cdot \{a\}, t; \{p\}_\alpha)\|$.

Ainsi, afin d'améliorer la prévision des autres états correspondant aux valeurs non étudiées des paramètres, un modèle d'ordre réduit est construit en sélectionnant les s premiers vecteurs propres empiriques $\left(\underline{\boldsymbol{\psi}}_k\right)_{k=1\ldots s}$ comme fonctions de forme du modèle d'ordre réduit. Plus la différence entre les nouvelles valeurs de paramètres et celles utilisées pour l'expansion de Karhunen-Loève est faible, meilleure sera la qualité du modèle d'ordre réduit. Mais cette méthode n'inclut pas d'adaptation d'un modèle d'ordre réduit connu à des modifications de certains paramètres du modèle détaillé.

1.1.3 Méthodes de séparation de variables

1.1.3.1 Une problématique ancienne

Les méthodes de réduction de modèle sont basées sur des techniques de séparation de variables. En effet, la recherche du champ inconnu $\underline{\boldsymbol{u}}$ est basée sur une technique de séparation de variables temps/espace, comme l'on peut le voir en (1.7). Il s'agit d'une problématique très ancienne : dès la seconde moitié du XVIIe siècle, le domaine mathématique de l'analyse numérique connut une avancée prodigieuse grâce aux travaux de Newton et de Leibniz en matière de calcul différentiel (les équations différentielles, les équations aux dérivées partielles,...). En 1797, Sylvestre-François Lacroix publie le "Traité du calcul différentiel et de calcul intégral" qui se veut une synthèse des travaux d'analyse du XVIIIe siècle avec un chapitre consacré à la séparation des variables dans les équations différentielles du premier

ordre. La méthode de séparation des variables constitue l'une des méthodes de résolution des équations différentielles partielles et ordinaires, lorsque l'algèbre permet de réécrire l'équation, de sorte que chacune des deux variables apparaisse dans un membre distinct de l'équation. Pour le premier ordre, l'équation différentielle du premier degré s'écrit de la forme :

$$Mdx + Ndy = 0 \tag{1.59}$$

Et le moyen de résoudre cette équation a été de chercher à séparer les variables, c'est à dire ramener l'équation (1.59) à la forme :

$$Xdx + Ydy = 0 \tag{1.60}$$

où X (resp. Y) est une fonction dépendant uniquement de x (resp. y). Différents auteurs se sont intéressés à la mise en oeuvre de ces techniques d'un point de vue numérique pour construire des méthodes de réduction de modèles. Contrairement aux travaux antérieurs, il ne s'agit plus alors d'aboutir à une solution analytique. Nous la considèrerons donc comme un outil mathématique de résolution des problèmes de dimensions élevées.

Considérons un champ de déplacement $\underline{\boldsymbol{u}}(x, y)$ fonctions des deux coordonnées $x \in \Omega_x$ et $y \in \Omega_y$, avec $\Omega = \Omega_x \times \Omega_y$.

Dans ce cas, la forme du problème différentiel (1.1) devient :

$$\mathcal{K}(\underline{\boldsymbol{u}}(x, y)) + \mathcal{L}(\underline{\boldsymbol{u}}(x, y)) = 0 \tag{1.61}$$

où $\mathcal{K}$ et $\mathcal{L}$ sont deux opérateurs différentiels linéaires représentant l'équation aux dérivées partielles qui relie le champ inconnu $\underline{\boldsymbol{u}}$ à ses dérivées respectivement en x et en y.

La formulation variationnelle (1.6) devient :

$$\int_\Omega \left(\mathcal{K}(\underline{\boldsymbol{u}}(x, y)) + \mathcal{L}(\underline{\boldsymbol{u}}(x, y))\right) \underline{\boldsymbol{u}}^*(x, y) dxdy \quad = \quad 0, \quad \forall \underline{\boldsymbol{u}} \in \mathcal{V} \tag{1.62}$$

La recherche de la solution de l'équation aux dérivées partielles (1.61) se fait sous la forme d'une somme de produits tensoriels. On définit le produit tensoriel suivant : pour $(r_n, s_n) \in \Omega_x \times \Omega_y$,

$$r_n \otimes s_n : \begin{cases} \Omega_x \times \Omega_y \to \Omega \\ \quad (x, y) \mapsto r_n(x) s_n(y) \end{cases} \tag{1.63}$$

Nous sommes conduits à écrire $\underline{\boldsymbol{u}}(x,y)$ sous la forme :

$$\underline{\boldsymbol{u}}(x,y) = \sum_{n\geq 1} r_n(x) \otimes s_n(y) \tag{1.64}$$

Un des premiers exemples d'approximation de ce type a été donné par Schmidt [Schmidt, 1908] qui considère l'approximation de fonctions de deux variables $f(x,y)$ par les formes bilinéaires $\sum_{i=1}^{m} u_i(x)v_i(y)$ dans $L_2([0,1]^2)$. Dans [Ojalvo, 1962], l'auteur présente une extension de la méthode de séparation de variables aux problèmes de conduction thermique avec source de chaleur, et aux conditions aux limites dépendant du temps. La modification de la méthode permet de transformer les problèmes généraux de ce type, en plusieurs problèmes relatifs aux phénomènes stationnaires et transitoires dont les techniques de résolution sont parfaitement connues. L'exemple présenté par l'auteur est l'équation de conduction de la chaleur en régime dynamique (T étant la température et a_t la diffusivité thermique).

$$\frac{\partial^2 T}{\partial x^2} = a_t^{-1} \frac{\partial T}{\partial t} \tag{1.65}$$

avec les conditions aux limites et conditions initiales :

$$\begin{aligned} T(0,t) &= f(t) \\ T(L,t) &= 0 \\ T(x,0) &= h(x) \end{aligned} \tag{1.66}$$

Pour résoudre le problème ((1.65)-(1.66)), la solution est supposée sous la représentation à variables séparées :

$$T(x,t) = \sum_{n} X_n(X)\Psi_n(t) + T_0(x)f(t) \tag{1.67}$$

où T_0 satisfait l'équation de conduction de la chaleur en régime stationnaire :

$$\frac{\partial^2 T_0}{\partial x^2} = 0 \tag{1.68}$$

1.1.3.2 Algorithme glouton

Les algorithmes gloutons consistent à minimiser localement une fonctionnelle par un produit tensoriel. La construction de la solution se fait par séparation de variables. Ainsi, l'algorithme glouton est la méthode qui permet de déterminer de manière itérative $r_n(x)$ et $s_n(y)$ tel que $\forall n$, $\sum_{n\geq 1} r_n \otimes s_n$ soit la meilleure approximation possible (dans un sens à définir) de la solution $\underline{\boldsymbol{u}}(x,y)$. L'algorithme consiste

- à déterminer les meilleures fonctions $R(x)$ et $S(y)$ à l'itération n qui vérifient (1.62) et telles que :

$$\underline{\boldsymbol{u}}\,(x,y) = \sum_{i=1}^{n-1} r_i(x)s_i(y) + R(x)S(y) \tag{1.69}$$

(les $r_{i,(i=1...n-1)}$ et $s_{i,(i=1...n-1)}$ étant connues des itérations précédentes.)
- à résoudre le problème de Galerkin sur la base $(r_1 \otimes s_1, ..., r_n \otimes s_n)$

Trois différents algorithmes glouton (Pure Greedy Algorithm ou PGA, Relaxed Greedy Algorithm ou RGA, et Orthogonal Greedy algorithm ou OGA) ont été étudiés par DeVore et Temlyakov ([DeVore and Temlyakov, 1996]), et l'analogue de ces algorithmes en formulation faible sont définis en [Temlyakov, 2000], afin de rendre ces algorithmes plus pratiques pour leur implémentation. Le Bris et al. présentent deux de ces algorithmes (PGA et OGA) dans [Le Bris et al., 2009] pour résoudre le problème classique et très simple d'une équation de Poisson avec condition de Dirichlet homogène sur les bords :

$$\text{Trouver } \underline{\boldsymbol{u}} \in H_0^1(\Omega) \text{ tel que } \begin{cases} -\Delta\underline{\boldsymbol{u}} = f \text{ sur } \Omega = \Omega_x \times \Omega_y \\ \underline{\boldsymbol{u}} = 0 \text{ sur } \partial\Omega \end{cases} \tag{1.70}$$

L'algorithme PGA (Algorithme 1) et sa variante OGA (Algorithme 2) sont utilisés pour résoudre ce problème.

Data: ;
$f_0 = f$;
BEGIN
$n = 1$
while $\|f_n\|_{H^{-1}(\Omega)\geq\varepsilon}$ **do**
1. Trouver $r_n \in H_0^1(\Omega_x)$ et $s_n \in H_0^1(\Omega_y)$ tel que;

$$(r_n, s_n) = \underset{(r,s)\in H_0^1(\Omega_x)\times H_0^1(\Omega_y)}{\arg\min} \left(\frac{1}{2}\int_\Omega |\nabla(r\otimes s)|^2 \, d\Omega - \int_\Omega f_{n-1} r\otimes s d\Omega \right) \tag{1.71}$$

2. $f_n = f_{n-1} + \Delta(r_n \otimes s_n)$;
3. $n = n + 1$;

end
END

Algorithm 1: Pure Greedy Algorithm.

Data: ;
$f_0^o = f$;
BEGIN
$n = 1$
while $\|f_n^o\|_{H^{-1}(\Omega)} \geq \varepsilon$ **do**
1. Trouver $r_n^o \in H_0^1(\Omega_x)$ et $s_n^o \in H_0^1(\Omega_y)$ tel que;

$$(r_n^o, s_n^o) = \underset{(r,s)\in H_0^1(\Omega_x)\times H_0^1(\Omega_y)}{\arg\min} \left(\frac{1}{2}\int_\Omega |\nabla(r\otimes s)|^2 \, d\Omega - \int_\Omega f_{n-1}^o r\otimes s d\Omega \right) \tag{1.72}$$

2. Résoudre le problème de Galerkin suivant, sur la base $(r_1^o \otimes s_1^o, ..., r_n^o \otimes s_n^o)$ *i.e* trouver $(\alpha_1, ..., \alpha_n) \in \mathbb{R}^n$ tel que :

$$(\alpha_1, ..., \alpha_n) = \underset{(\beta_1,...,\beta_n)\in\mathbb{R}^n}{\arg\min} \left(\frac{1}{2}\int_\Omega \left| \nabla \left(\sum_{k=1}^n \beta_k r_k^o \otimes s_k^o \right) \right|^2 d\Omega - \int_\Omega f \sum_{k=1}^n \beta_k r_k^o \otimes s_k^o d\Omega \right) \tag{1.73}$$

3. $f_n^o = f + \Delta\left(\sum_{k=1}^n \alpha_k r_k^o \otimes s_k^o\right)$;
4. $n = n + 1$;

end
END

Algorithm 2: Orthogonal Greedy Algorithm.

On retrouve ce type d'algorithme glouton dans la méthode PGD (Proper Generalized Decomposition) que l'on va décrire dans la section suivante.

1.1.3.3 Proper Generalized Decomposition

Il est bien connu qu'en utilisant une méthode de type Éléments Finis, le nombre de degrés de liberté d'un modèle augmente exponentiellement avec la dimension de l'espace. Ammar et ses collègues [Ammar et al., 2006a] donnent un exemple concrêt qui permet de comprendre ce problème de la dimensionnalité, en prenant un modèle défini sur un hypercube dans un espace de dimension D, $\Omega =]-L, L[^D$. La plupart des méthodes numériques (entre autres celle qui nous intéresse, la méthode des Éléments Finis) repose sur un découpage de l'espace selon un maillage, pour discrétiser le modèle. Supposons que ce maillage comporte N noeuds dans chaque direction, alors le nombre total de noeuds est de N^D. Si l'on suppose que $N \approx 10$, (un maillage extrêmement grossier) et $D \approx 80$ (ce qui est même plus faible que les dimensions requises en mécanique statistique ou quantique), alors le nombre de noeuds du maillage atteint la valeur astronomique de 10^{80} !
Une des méthodes qui permet d'alléger le problème est une extension de la méthode POD. En effet, quasiment pendant la même période que Sirovich, en 1985, Ladevèze a proposé une méthode concurrentielle de décomposition dédiée aux modèles élastoplastiques, la Décomposition en Chargement Radial (RLD) [Ladevèze, 1985] construite par la méthode LATIN. Nous reviendrons sur cette méthode LATIN dans la section suivante. Les similarités entre la RLD et la POD ont été étudiées en [Ryckelynck, 2002].
Cette méthode repose sur la construction a priori d'une décomposition en variables séparées de la solution, qui exploite la structure produit tensoriel de l'espace de travail. Ce type de représentation n'est pas nouveau. Dans le contexte de la mécanique numérique, c'est la méthode RLD qui était appliquée pour séparer les variable d'espace et de temps dans des modèles thermomécaniques.
Lorsque les équations à résoudre peuvent être formulées sous une forme séparée, de façon exacte ou de manière approchée, la Décomposition en Chargement Radial conduit à la Proper Generalized Decomposition (PGD), en factorisant la forme faible du problème à résoudre. Les premiers travaux réalisés sur ce type d'approche ont été réalisés par Ammar et Chinesta [Ammar et al., 2006b] [Ammar et al., 2006a].

C'est cette méthode RLD, baptisée récemment Proper Generalized Decomposition (PGD) que nous allons présenter dans cette section.

La méthode PGD utilise un algorithme glouton que l'on a décrit en section 1.1.3.2, avec une hypothèse supplémentaire de séparabilité des variables et des équations, ce qui permet d'aboutir à des sommes de produits d'intégrales monodimensionnelles. Pour faciliter la lecture, la méthode sera formulée pour

$N = 2$. Ces deux coordonnées sont $x \in \Omega_x$ et $y \in \Omega_y$ avec $\Omega = \Omega_x \times \Omega_y$. Comme nous l'avons vu dans l'algorithme glouton, la correction que l'on veut construire est cherchée sous la forme $R(x)S(y)$ et la solution à l'itération n est donc de la forme $\underline{\boldsymbol{u}}\,(x,y) = \sum_{i=1}^{n-1} r_i(x)s_i(y) + R(x)S(y)$. $R(x)$ et $S(y)$ sont calculées par une méthode itérative (méthode du point fixe, par exemple). La fonction test à injecter dans la formulation variationnelle du problème s'écrit sous la forme $\underline{\boldsymbol{u}}^*(x,y) = R^*(x)S(y)$. si l'on suppose $S(y)$ connue et que l'on cherche à calculer $R(x)$, de la manière suivante.

Algorithme PGD :

- Étape 1 :
  L'injection de la fonction test

$$\underline{\boldsymbol{u}}^*(x,y) = R^*(x)S(y) \tag{1.74}$$

et de la solution

$$\underline{\boldsymbol{u}}\,(x,y) = \sum_{i=1}^{n-1} r_i(x)s_i(y) + R(x)S(y) \tag{1.75}$$

dans la formulation variationnelle (1.6), et la définition des opérateurs différentiels $\mathcal{K}$ et $\mathcal{L}$ mènent à la formulation suivante :

$$\begin{aligned} &\int_\Omega R^* \cdot S \cdot (S \cdot \mathcal{K}(R) + R \cdot \mathcal{L}(S))\, dxdy = \\ &-\int_\Omega R^* \cdot S \cdot \Big(\sum_{i=1}^{i=n-1} (s_i \cdot \mathcal{K}(r_i) + r_i \cdot \mathcal{L}(s_i)) \Big) dxdy. \end{aligned} \tag{1.76}$$

$S(y)$ étant connue et les équations étant séparables, nous nous retrouvons à intégrer l'équation (1.76) sur x seulement :

$$\int_{\Omega_x} R^* \cdot (\alpha^y \mathcal{K}(R) + \beta^y R) dx = -\int_{\Omega_x} R^* \cdot \Big(\sum_{i=1}^{i=n-1} (\alpha_i^y \mathcal{K}(r_i) + \beta_i^y s_i) \Big) dx, \tag{1.77}$$

où

$$\begin{aligned}
\alpha^y &= \int_{\Omega_y} S^2 dy \\
\alpha_i^y &= \int_{\Omega_y} S \cdot s_i dy \\
\beta^y &= \int_{\Omega_y} S \cdot \mathcal{L}(S) dy \\
\beta_i^y &= \int_{\Omega_y} S \cdot \mathcal{L}(s_i)\, dy
\end{aligned} \tag{1.78}$$

- Étape 2 :
  On applique un algorithme glouton pour rechercher les fonctions $R(x)$ et $S(y)$: $R(x)$ étant maintenant connue, on injecte $\underline{\boldsymbol{u}}^*(x,y) = R(x)S^*(y)$ dans la formulation variationelle et l'on peut calculer de la même manière $S(y)$.
  $S(y)$ à nouveau connue, on recommence l'étape 1, jusqu'à convergence de l'algorithme.

Comme nous l'avons expliqué précédemment, cette méthode est particulièrement efficace pour résoudre des problèmes multidimensionnels, le nombre de degrés de libertés augmentant exponentiellement avec la dimension de l'espace. La simulation multidimensionnelle semble être une approche séduisante grâce aux récents travaux d'Ammar et Chinesta. L'algorithme proposé dans [Ammar et al., 2006a] est relatif à l'algorithme OGA : il consiste à remplacer la procédure d'optimisation (1.72) par les équations d'Euler Lagrange associées. Tandis que le sujet de cet article porte sur la solution permamente d'EDP elliptiques multidimensionnelles définies sur des espaces de dimension N, un second article des mêmes auteurs [Ammar et al., 2006b] étend le premier au traitement des solutions transitoires des EDP paraboliques multidimensionnelles. Ces deux papiers utilisent la même technique basée sur l'utilisation de bases réduites dans une procédure adaptative avec séparation de variables. En effet, les fonctions R et S que l'on rajoute ne sont pas connues a priori, mais sont construites en cours de procédure. Nous allons décrire les différentes étapes de cette technique en se basant sur le problème relativement simple de l'équation de Poisson en dimension N.

$$\Delta T = -f(x_1, x_2, ..., x_N), \quad (x_1, x_2, ..., x_N) \in \Omega =]-L, +L[^N \tag{1.79}$$

Comme nous l'avons déjà expliqué, la solution d'un tel problème peut être

approchée par un nombre fini de fonctions d'approximation, i.e

$$T(x_1, x_2, ..., x_N) = \sum_{j=1}^{Q} \alpha_j \prod_{k=1}^{N} F_{kj}(x_k) \tag{1.80}$$

où $F_{kj}(x_k)$ est la $j^{ième}$ fonction de base. Le schéma numérique proposé par Ammar et al. consiste en une procédure itérative qui résoud à chaque itération n les 3 étapes suivantes :

- Étape 1 : Projection de la solution sur une base discrète.
  Comme proposé dans l'algorithme glouton, T et T^* sont introduits sous la forme suivante :

$$\begin{aligned} T(x_1, x_2, ..., x_N) &= \sum_{j=1}^{n} \alpha_j \prod_{k=1}^{N} F_{kj}(x_k), \\ T^*(x_1, x_2, ..., x_N) &= \sum_{j=1}^{n} \alpha_j^* \prod_{k=1}^{N} F_{kj}(x_k) \end{aligned} \tag{1.81}$$

  La formulation variationnelle correspondant au problème (1.79) implique alors un produit de N fonctions, chacune étant définie sur une dimension différente. Notons $\prod_{k=1}^{N} g_k(x_k)$ une de ces fonctions à intégrer. L'intégrale sur Ω de la formulation variationnelle devient un produit d'intégrales monodimensionnelles, selon :

$$\int_\Omega \prod_{k=1}^{N} g_k(x_k) d\Omega = \prod_{k=1}^{N} \int_{-L}^{L} g_k(x_k) dx_k \tag{1.82}$$

  La résolution d'un système linéaire de taille $n \times n$ permet de calculer les coefficients α_j, et donc la solution T.
- Étape 2 : Vérification de la convergence grâce au résidu

$$R = \frac{\sqrt{\int_\Omega \left(\Delta T + f(x_1, x_2, ..., x_N)\right)^2}}{\|T\|^2} \tag{1.83}$$

  Si $R > \epsilon$, arrêt de la procédure itérative, et la solution est donnée par (1.81).
- Étape 3 : Enrichissement de la base.
  Connaissant les coefficients α_i, la base peut être enrichie en ajoutant

la nouvelle fonction $\prod_{k=1}^{N} F_{k(n+1)}(x_k)$. Pour cela, on injecte une fonction test

$$T^*(x_1, x_2, ..., x_N) = R_1^*(x_1)R_2(x_2)...R_N(x_N)+...+R_1(x_1)R_2(x_2)...R_N^*(x_N) \tag{1.84}$$

dans la formulation variationnelle, ce qui mène à un problème variationnel non-linéaire dont la solution permet de calculer les N fonctions $R_k(x_k)$. (Il s'agit de reprendre l'algorithme PGD décrit au début de la section 1.1.3.3).
Chaque fonction $F_{k(n+1)}(x_k)$ est alors obtenue en normalisant la fonction associée R_k.

Les résultats numériques ont été concluants pour les deux articles cités en terme de qualité, les temps de calculs n'étant pas comparés. Dans tous les cas traités, les conditions aux limites sont homogènes. Mais le traitement de problèmes aux conditions aux limites non homogènes a été étudié par Gonzalez et al. [Gonzalez et al., 2010] sur des équations de Laplace. Un simple changement de variable permet de se rapporter au cas avec conditions homogènes. Ils traitent également des géométries complexes (d'autres domaines que des hypercubes) : la stratégie de résolution se base sur la méthode des fonctions caractéristiques.

Un autre exemple d'analyse multidimensionnelle utilisant la méthode PGD est présenté par Pruliere et al. [Prulière et al., 2010], pour une approche multidimensionnelle dans laquelle tous les cas seront considérés simultanément. La méthode est illustrée dans le cadre de la résolution paramétrique de l'équation de transfert de chaleur :

$$\frac{\partial \underline{\boldsymbol{u}}}{\partial t} - k\Delta\underline{\boldsymbol{u}} - f = 0, \tag{1.85}$$

où les variables sont l'espace $\underline{\boldsymbol{x}} \in \Omega$, le temps $t \in I$ et la diffusivité $k \in \mathcal{K}$. La solution est recherchée sous la forme :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{x}}\,, t, k) \approx \sum_{i=1}^{N} X_i(\underline{\boldsymbol{x}}\,) \cdot T_i(t) \cdot K_i(k). \tag{1.86}$$

où $\underline{\boldsymbol{x}}$ peut encore être séparé en 3 variables : (x, y, z). De même que pour les exemples présentés précédemment, le produit $R(\underline{\boldsymbol{x}}\,) \cdot S(t) \cdot W(k)$ est cherché de manière itérative.

Dans cet exemple, une des coordonnées est le temps. L'algorithme est appliqué sur tout l'intervalle de temps : la méthode est non-incrémentale en

temps.

Évidemment, les exemples d'applications de ce type de représentation sont nombreux, puisque cette technique est une bonne stratégie quand la dimension du modèle augmente. En effet, la taille du problème croît linéairement avec la dimension de l'espace, au lieu d'exponentiellement avec une méthode numérique classique.
Mais cette technique n'est pas pour autant une stratégie universelle pour résoudre tout type d'EDP multidimensionnelles. De plus, l'efficacité de la méthode nécessite la séparabilité de tous les champs impliqués dans le modèle. Mais cette hypothèse de séparabilité n'est pas toujours possible, ainsi, en section 4, nous proposerons de compléter la méthode d'Hyper Réduction proposée en [Ryckelynck, 2005] et [Ryckelynck, 2009] (détaillée en section 1.3.2) pour les méthodes de séparation de variables.
Tandis que la méthode PGD est basée sur la séparation de variables, dans la section suivante, nous présentons la méthode LATIN, basée sur la séparation de la nature des équations.

1.1.3.4 Méthode LATIN

La méthode LATIN [Ladevèze, 1985], dans ses grandes lignes, comporte deux principales étapes.

1. Partitionnage du domaine espace-temps. La structure est décomposée en un ensemble de sous-structures et d'interfaces. L'intervalle de temps est divisé en sous-intervalles.
2. La séparation en deux groupes des équations à résoudre en distinguant :
   - l'espace des solutions des équations globales linéaires (étape globale),
   - l'espace des solutions des équations non-linéaires locales en chaque point d'intégration (étape locale).

Comme la méthode PGD, qui est un cas particulier de la méthode LATIN, la méthode est non incrémentale itérative. À chaque itération, on obtient une approximation de la réponse sur la structure complète, sur l'intervalle de temps entier. L'étape globale consiste à résoudre alternativement à chaque itération deux types de problèmes :

- le problème macro homogénéisé défini sur l'espace temps complet,
- une famille de problèmes linéaires (problèmes micro) indépendants par sous-structure.

Un algorithme alliant méthode LATIN et méthode de type POD a été proposé par Ladevèze et Nouy pour résoudre des problèmes multi-échelles [Ladevèze and Nouy, 2003]. Dans cet article, une nouvelle stratégie de calcul

multi-échelle basée sur la méthode LATIN a été proposée pour l'analyse de structures dont le comportement est décrit à une échelle fine, tant en temps qu'en espace. La stratégie de décomposition de domaine évoquée ci-dessus conduit à la résolution d'un grand nombre de problèmes d'évolution microscopiques indépendants par structures à chaque itération. La représentation RLD décrite en section 1.1.3.3 a été introduite dans [Arzt, 1994] pour des problèmes élastoviscoplastiques.
La représentation à variables séparées intervient dans la recherche de la solution des équations globales linéaires : la solution du problème micro est approximé par une somme de fonctions radiales, qui sont des produits de fonctions scalaires dépendant du temps par des fonctions d'espace. Ils vont donc chercher la solution du problème d'évolution sous la forme

$$f(\underline{\boldsymbol{x}}\,,t) = \sum_{i=1}^{m} \lambda_i(t)\Lambda_i(\underline{\boldsymbol{x}}\,)$$

où λ_i et Λ_i sont connues a priori. Pour résoudre le micro-problème (reformulé en problème de minimisation) sur l'espace des fonctions radiales, un schéma itératif est utilisé, consistant à minimiser alternativement sur les fonctions de temps, puis les fonctions d'espace. En pratique, pour chaque variable, une seule fonction est calculée en même temps, et le schéma itératif est stoppé après deux itérations.
La minimisation sur les fonctions du temps conduit à un simple système différentiel d'équations, tandis que minimiser sur les fonctions d'espace mène à un problème spatial. Ce problème spatial et le système différentiel étant découplés, le coût pour résoudre l'étape linéaire est quasiment indépendant du schéma d'intégration en temps. Ainsi, pour le même coût, il est possible d'utiliser un schéma d'intégration implicite robuste sur un maillage en temps très fin. Au fur et à mesure des itérations, les fonctions d'espace construites constituent une base réutilisée pour les itérations suivantes. Chaque itération commence par une étape préliminaire dont le but est de trouver l'ensemble des fonctions temporelles associées à l'ensemble des fonctions spatiales de la base.
Ainsi, à l'itération n, la résolution du microproblème se fait en deux étapes, que nous avons déjà évoquées :

- étape de prédiction : la résolution de l'étape préliminaire permet de déterminer une solution approchée pour un coût assez faible,
- enrichissement de la base : correction si la qualité n'est pas assez bonne.

Il s'agit donc d'une sorte de POD adaptative non incrémentale.
Un très petit nombre de fonctions étant nécessaire pour approximer correctement la solution définie sur le domaine entier espace-temps, le temps

de calcul de l'étape linéaire, qui était l'étape la plus coûteuse, est réduit considérablement. L'efficacité exceptionnelle de cette méthode provient donc de l'utilisation d'approximations RLD.

Ainsi, les algorithmes LATIN peuvent contrôler la qualité de la représentation en variables séparées en utilisant un schéma non incrémental. Il est également possible d'adapter le maillage utilisé pour représenter les champs de la représentation RLD au cours de la convergence de cette représentation en introduisant différents indicateurs d'erreurs [Pelle and Ryckelynck, 2000].
Comme Ryckelynck le montre en [Ryckelynck, 2002], la méthode LATIN peut être considérée comme une méthode a priori de réduction de modèle qui peut être optimisée grâce à l'expansion de Karhunen-Loève. Ces méthodes de réduction de modèles sont utilisés dans de nombreux domaines, et à présent, nous allons donner quelques exemples d'application de ces méthodes en science des matériaux.

1.2 Réduction de modèles dans le cadre de la science des matériaux et méthodes apparentées

1.2.1 Réduction de modèles de problèmes avec microstructure

Une des nombreuses utilisations de la méthode POD en science des matériaux est présentée dans l'article [Ganapathysubramanian and Zabaras, 2004]. Le but des auteurs est de développer des techniques de contrôle des propriétés mécaniques de matériaux polycristaux. La texture cristalline est décrite quantativement par des Fonctions de Distribution des Orientations cristallines (ODF) $\mathcal{A}(\underline{\boldsymbol{r}}, t)$, ($\underline{\boldsymbol{r}}$ représentant l'orientation du cristal). La microstructure comportant de nombreux degrés de liberté, la réduction de modèle est introduite sur la microéchelle, afin de modéliser l'évolution de la microstructure pendant un processus de déformation plastique.
L'équation sur laquelle se penchent les auteurs est la suivante :

$$\frac{\partial \mathcal{A}(\underline{\boldsymbol{r}}, t)}{\partial t} + \nabla \mathcal{A}(\underline{\boldsymbol{r}}, t).\underline{\boldsymbol{v}}(\underline{\boldsymbol{r}}, t) + \mathcal{A}(\underline{\boldsymbol{r}}, t)\nabla.\underline{\boldsymbol{v}}(\underline{\boldsymbol{r}}, t) = 0. \tag{1.87}$$

où $\underline{\boldsymbol{v}}(\underline{\boldsymbol{r}}, t)$ est la vitesse de réorientation Eulérienne.
La formulation Streamline Upwind Petrov-Galerkin combinée à un modèle de

capture de chocs (introduisant un paramètre de capture de chocs ε) permet de passer de l'équation (1.87) à :

$$\int_{\mathbb{R}} \left\{ \frac{\partial \mathcal{A}}{\partial t} + \nabla \mathcal{A}.\underline{\boldsymbol{v}} + \mathcal{A}\nabla.\underline{\boldsymbol{v}} \right\} \eta dv + \int_{\mathbb{R}} \nabla.(\varepsilon \nabla \mathcal{A})\varphi dv = 0. \tag{1.88}$$

Il s'agit d'approcher les ODF $\mathcal{A}(\underline{\boldsymbol{r}}, t)$ par :

$$\mathcal{A}(\underline{\boldsymbol{r}}, t) = \sum_{i=1}^{z} a_i(t)\phi_i(\underline{\boldsymbol{r}}) \tag{1.89}$$

$$a_i(t) = \int_{\mathbb{R}} \mathcal{A}(\underline{\boldsymbol{r}}, t)\phi_i(\underline{\boldsymbol{r}})dv \tag{1.90}$$

où les fonctions de bases optimales $\phi_i(\underline{\boldsymbol{r}})$ pour la représentation des ODF sont générées en utilisant la méthode Snapshot POD.

ϕ est exprimée comme une combinaison linéaire des réalisations $\underline{\boldsymbol{u}}$:

$$\phi_k = \sum_{i=1}^{N} u_i^j \mathcal{A}^i \tag{1.91}$$

Le problème aux valeurs propres à résoudre est le suivant :

$$[C]\,[U] = [\Lambda]\,[U]\,, \tag{1.92}$$

où $[C]$ est la matrice de corrélation spatiale définie par :

$$C_{ij} = \frac{1}{N} \int_{\mathbb{R}} \mathcal{A}^i(\underline{\boldsymbol{r}})\mathcal{A}^j(\underline{\boldsymbol{r}})dv \tag{1.93}$$

Après avoir évalué les modes ϕ, l'injection de (1.90) dans (1.88) conduit à une équation différentielle ordinaire du type :

$$\{\dot{a}\} = [B]\,\{a\} + \underline{\boldsymbol{b}} \tag{1.94}$$

où $\{a\}$ permet de déterminer les ODF.
Ces ODF obtenues grâce à la réduction de modèle seront comparées aux ODF obtenues par la méthode Éléments Finis. Les résultats sont globalement satisfaisants, mais les auteurs arrivent à la conclusion qu'une sélection adaptative des fonctions de base pourrait conduire à une performance optimale de l'algorithme.

Les travaux d'Ammar et ses collègues [Ammar et al., 2006c] portent quant à eux sur l'extension des méthodes de réduction de modèles à un modèle cinétique relatif à un modèle continu de polymère : le modèle FENE est un modèle élastique non-linéaire de façon finie extensible. Un formalisme de Fokker-Planck est utilisé pour décrire le modèle. L'équation de Fokker-Planck étant définie sur un espace multi-dimensionnel, le nombre de degrés de libertés devient vite exorbitant. C'est pourquoi les auteurs se sont intéressés à une méthode de réduction a priori (A Priori Reduction) proposée par Ryckelynck en 2005 [Ryckelynck, 2005], où la construction des fonctions d'approximation est faite en combinant une réduction de la base par POD, et un enrichissement de cette même base en utilisant des sous-espaces de Krylov. Les détails de cette méthode APR peuvent être trouvés dans la section 1.3.2. Cette méthode a été appliquée à la résolution de modèles viscoélastiques FEBE en 1D, 2D et 3D. En 1D, les auteurs ont réussi à réduire considérablement le nombre de fonctions, en passant de 316 fonctions de formes Éléments Finis à seulement 6 fonctions de base nécessaires pour représenter l'évolution de la solution en utilisant la méthode APR. Avec d'autres paramètres, le gain est encore plus important (ils passent de 706 fonctions de formes Éléments Finis à seulement 11 fonctions de base). En 2D et 3D, le nombre de fonctions d'approximation est le même : seule une dizaine est nécessaire pour obtenir d'excellents résultats au niveau qualitatif.

Récemment, Ryckelynck, fondateur de la méthode APHR, a étendu ses travaux afin de réduire encore plus les temps de calculs, notamment dans le cadre général pour les problèmes à variables internes en mécanique. La principale différence par rapport à la méthode APR est l'extension de l'Hyper Réduction aux modèles à variables internes [Ryckelynck, 2009], sur laquelle nous reviendrons en section 1.3.2.2. Cette méthode multi-niveau appelée A Priori Hyper Reduction (APHR) peut être considérée comme un nouveau solveur efficace, et une extension de ce solveur a été proposée pour le calcul parallèle, en utilisant un solveur parallèle FETI [Ryckelynck and Missoum-Benziane, 2010]. La méthode a été testée sur un problème classique et assez simple d'une plaque carrée trouée constituée d'un matériau élastoviscoplastique (le problème pour le calcul parallèle est une plaque percée de 16 trous). L'efficacité de 5 différentes méthodes est comparée : Éléments Finis, Snapshot POD, APR et la méthode APHR qu'il propose (une précise, et une autre qui l'est un peu moins, mais qui est plus rapide). Et ce, en utilisant une suite de 3 simulations différant les unes des autres par une variation de deux paramètres : un paramètre matériau c, et le rayon du trou r. L'Hyper Réduction permettant un meilleur gain en temps de calcul, l'auteur ne se contente plus de comparer le nombre de vecteurs de bases obtenus, mais le temps CPU

obtenu pour effectuer les 3 simulations. Le meilleur gain en temps de calcul obtenu est de 97% par rapport à une simulation par Éléments Finis grâce au calcul parallèle, et de 76% en séquentiel. Dans les 2 cas, les simulations ont été effectuées avec la méthode APHR, ce qui prouve ainsi son efficacité, le niveau d'erreur par rapport aux simulations Éléments Finis étant inférieur à 10%.

On peut trouver dans un autre article du même auteur, [Ryckelynck, 2010], une application de cette méthode particulière de réduction de modèle APHR à un modèle réel en science des matériaux en plasticité cristalline pour le calcul d'agrégats. La simulation d'agrégats polycristallins est classique en science des matériaux, par exemple pour obtenir les contraintes locales et les champs de déformations [Musienko et al., 2007]. Pour ce genre de simulation le maillage Éléments Finis est constitué de plusieurs centaines de grains. Pour chaque grain, la loi de comportement utilisée est présentée ci-après afin d'illustrer que la méthode peut être appliquée à des lois de comportement très complexes et non séparables.
Cette méthode a permis de diviser par 361 le temps de calcul de la simulation, mais a également permis une économie d'énergie considérable : un gain de 5769. Cette méthode est donc un premier pas vers la simulation numérique écologique !

1.2.1.1 Méthode R3M

Le calcul des propriétés effectives des matériaux hétérogènes hyperélastiques en grandes déformations est crucial dans certaines applications, telles que l'étude des tissus biologiques ou des mousses polymères. Dans leur étude, [Yvonnet and He, 2007], Yvonnet et He présentent une méthode multi-échelle pour l'homogénéisation non linéaire des matériaux hétérogènes hyperélastiques, en grandes déformations : la méthode R3M (Reduced Model Multiscale Method). Le processus se déroule en trois étapes :

- Analyse numérique multi-niveau :
  (i) Problème macro (sur la structure entière) : une procédure de type Éléments Finis multi-niveaux (FE2), telle que chaque point matériel de la structure macroscopique (les points de Gauss) soit associé à un modèle Éléments Finis de Volume Élémentaire Représentatif (VER).
  (ii) Problème micro (sur un VER en chaque point d'intégration macro) : Les non-linéarités provenant de la micro-échelle, chaque problème non-linéaire associé à l'échelle microscopique est remplacé par un problème de taille réduite n'impliquant qu'un faible nombre de degrés de liberté.
- Pré-calcul sur le VER pour obtenir la base réduite.

- Résolution du problème micro en utilisant la POD.

Nous allons détailler ces étapes (principalement celles qui nous intéressent (c'est-à-dire les deux dernières), sur un problème de matériau hétérogène hyperélastique. Soit un solide D-dimensionnel soumis à un chargement quasi-statique dépendant du temps, sur un intervalle de temps $[O,T]$ discrétisé en M instants $\{t_1,...,t_M\}$. Soit Ω^0 un ouvert de $\mathbb{R}^D$, et $\partial\Omega^0$ le bord de ce domaine.

- **Analyse numérique multi-niveau :**
  (i) *Problème macroscopique*
  Soit $\underline{\boldsymbol{u}}\,(t) \in H^1(\Omega^0,t)$ le champ de déplacements macroscopiques, $\underset{\sim}{\bar{\boldsymbol{F}}}$ le tenseur gradient des déformations macroscopiques, et $\underset{\sim}{\bar{\boldsymbol{P}}}$ le tenseur des contraintes macroscopiques.
  Le modèle est défini par 4 différents chargements contenus dans des matrices $\underset{\sim}{\boldsymbol{F}}^*_{\alpha,(\alpha=1,...,4)}$.
  Le problème à résoudre est le suivant :

$$\nabla.\underset{\sim}{\bar{\boldsymbol{P}}} + \underset{\sim}{\bar{\boldsymbol{B}}} = 0 \tag{1.95}$$

  où $\underset{\sim}{\bar{\boldsymbol{B}}}$ est une densité de force volumique. La formulation faible de ce problème est la suivante :
  Trouver $\underline{\boldsymbol{u}} \in H^1(\Omega^0)$ vérifiant les conditions aux limites tel que, $\forall t \in [0,T]$,

$$\int_{\Omega^0} \underset{\sim}{\bar{\boldsymbol{P}}}(t) : \nabla_X(\partial\underline{\boldsymbol{u}})d\Omega = \int_{\Omega^0} \underset{\sim}{\bar{\boldsymbol{B}}}.\partial\underline{\boldsymbol{u}}\,d\Omega + \int_{\partial\Omega^0_\sigma} \underline{\bar{\boldsymbol{t}}}\,.\partial\underline{\boldsymbol{u}}\,d\Omega \qquad \forall\partial\underline{\boldsymbol{u}} \in H^1_0(\Omega^0) \tag{1.96}$$

  (ii) *Problème microscopique*

  Soit Ω^0_μ un VER à la microéchelle voisin d'un point macro $\underline{\boldsymbol{X}}$.
  La formulation faible de ce microproblème sur Ω^0_μ est la suivante :
  Trouver $\underline{\boldsymbol{u}}_\mu(t) \in H^1(\Omega^0_\mu)$ vérifiant les conditions aux limites tel que, $\forall t \in [0,T]$,

$$\int_{\Omega^0_\mu} \underset{\sim}{\boldsymbol{P}}(t) : \nabla_X(\partial\underline{\boldsymbol{u}})d\Omega = \int_{\Omega^0_\mu} \underset{\sim}{\boldsymbol{B}}.\partial\underline{\boldsymbol{u}}\,d\Omega + \int_{\partial\Omega^0_{\mu\sigma}} \underline{\boldsymbol{t}}\,.\partial\underline{\boldsymbol{u}}\,d\Omega \qquad \forall\partial\underline{\boldsymbol{u}} \in H^1_0(\Omega^0_\mu) \tag{1.97}$$

- **Pré-calcul sur le VER pour obtenir la base réduite**
  Soit $\{q\}_i$ le vecteur de dimension $N \times D$ formé par les composantes des

déplacements aux N points du solide à l'instant $t_i \in [0, T]$.
Le processus de calcul de la base réduite est le suivant (Algorithme 3).

BEGIN
1. **for** c **do**
| h
end
aque cas de chargement $\alpha = 1 \rightarrow 4$ 1.1 Résoudre, $\forall t = t_1, ..., t_M$, le problème (1.97);
1.2 Calculer $\{\bar{q}\} = \frac{1}{M} \sum_{i=1}^{M} \{q\}(t_i)$;
1.3 Concaténer les vecteurs colonnes centrés $\{q\}_i - \{\bar{q}\}$ pour former la matrice $[U]_\alpha = \left[\{q\}(t_1) - \{\bar{q}\}; ...; \{q\}(t_M) - \{\bar{q}\}\right]$;
2. Concaténer les matrices $[U]_\alpha$ pour former la matrice $[V] = [[U]_1; ...; [U]_4]$;
3. Construire la matrice de covariance $[Q] = [V][V]^T$ (il s'agit donc d'une méthode de POD classique);
4. Résoudre le problème aux valeurs propres $[Q]\underline{\boldsymbol{\psi}}_k = \lambda_k \underline{\boldsymbol{\psi}}_k$;
5. Construire la base $[\Psi] = \left[\underline{\boldsymbol{\psi}}_1, ..., \underline{\boldsymbol{\psi}}_s\right]$ où $s < N \times D$ est choisi selon le critère (1.43);
END

Algorithm 3: Calcul de la base réduite.

- **Résolution du problème micro en utilisant la POD**
La discrétisation Galerkin Éléments Finis du problème (1.97) mène au système discret d'équations :

$$[K]_\mu^j \Delta\{q\}^{j+1} = \underline{\boldsymbol{f}}_{ext(\mu)} - \underline{\boldsymbol{f}}_{int}^{\,j}(\{q\}_\mu) \tag{1.98}$$

où $[K]$ est la matrice tangente, $\underline{\boldsymbol{f}}_{ext}$ et $\underline{\boldsymbol{f}}_{int}$ sont respectivement les vecteurs forces externes et internes. L'indice supérieur j représente l'indice d'itération. La réduction du système (1.98) se fait grâce à la POD et utilise donc les vecteurs de base réduite $\underline{\boldsymbol{\psi}}_k$ définis dans l'algorithme 3. On peut alors écrire :

$$\Delta\{q\}^{j+1} = \sum_{k=1}^{s} \underline{\boldsymbol{\psi}}_k \Delta a_k^{j+1} \tag{1.99}$$

où a_k sont les variables réduites. L'introduction de (1.99) dans (1.98), suivi d'une multiplication par $[\Psi]^T$ donne le système à résoudre :

$$[\Psi]^T [K]_\mu^j [\Psi] \Delta a^{j+1} = [\Psi]^T \left[\underline{\boldsymbol{f}}_{ext(\mu)} - \underline{\boldsymbol{f}}_{int}^{\,j}(\{q\}_\mu)\right] \tag{1.100}$$

Ce système résolu, les variables réduites peuvent être actualisées :

$$a^{j+1} = a^j + \Delta a^{j+1} \tag{1.101}$$

Ainsi, la méthode R3M est une méthode Éléments Finis multi-niveaux utilisée en tandem avec une méthode de réduction de modèle de type POD pour alléger les coûts liés aux nombreux problèmes non linéaires qu'il est nécessaire de résoudre aux points de Gauss.
Toutefois, le choix du pré-calcul utilisé pour construire la base est crucial pour obtenir une solution précise, alors qu'avec des méthodes dites a priori (c'est à dire qui ne nécessitent pas de connaissance au préalable du système physique), il n'y a pas besoin de ce pré calcul. Ces méthodes seront décrites en section 1.3.
L'auteur précise également qu'une adaptation de la base serait plus judicieuse.
Cette technique R3M a également été utilisée par Monteiro et al. [Monteiro et al., 2008] pour résoudre des problèmes de conduction, thermiques et électriques, fortement non linéaires, dans des structures faites de matériaux hétérogenes périodiques.

1.2.1.2 La méthode NTFA

La méthode NTFA (Nonuniform Transformation Field Analysis) proposée par Michel et al. [Michel and Suquet, 2003] permet de prédire le comportement de matériaux hétérogènes non-linéaires. Le lecteur pourra trouver la description en détail de cette méthode dans [Michel and Suquet, 2004].
Il ne s'agit pas à proprement parler d'une méthode de réduction d'ordre de modèle, puisque les conditions d'équilibre du modèle détaillé sont rigoureusement vérifiées. Il s'agit de choisir une représentation en base réduite des déformations irréversibles et de lui associer une formulation de la relation de comportement à une échelle macroscopique.
La structure de ces matériaux étant multiéchelle, en chaque point de la structure macroscopique est associée une structure microscopique, celle-ci étant fortement hétérogène. Il s'agit de tenir compte de la non-uniformité du champ de déformation inélastique $\varepsilon^p(\underline{\boldsymbol{x}}\,,t)$. Ainsi, le point clef de la NTFA est d'approximer $\varepsilon^p(\underline{\boldsymbol{x}}\,,t)$ en utilisant une base de dimension N de champs spatiaux hétérogènes (des modes inélastiques) $\underline{\boldsymbol{\mu}}^{(\alpha)}(\underline{\boldsymbol{x}}\,)$ et des coefficients dépendant du temps $\xi_\alpha(t)$ de la manière suivante :

$$\varepsilon(\underline{\boldsymbol{x}}\,,t) = \sum_{\alpha=1}^{N} \xi_\alpha(t)\underline{\boldsymbol{\mu}}^{(\alpha)}(\underline{\boldsymbol{x}}\,) \tag{1.102}$$

Les auteurs cherchent à réduire le nombre de variables internes. En effet, celui-ci est très important puisque la microstructure est divisée en N phases homogènes. Ce sont donc les variables internes associées à la plasticité ou la

viscoplasticité qui sont approximées.
On peut considérer ce type d'approche comme appartenant au groupe des algorithmes exploitant des bases réduites.
Cette méthode a été appliquée par Fritzen et Böhlke [Fritzen and Böhlke, 2010] pour la première fois à un problème 3D de composite à matrice métallique. Elle a l'avantage d'être très efficace puisqu'il n'y a besoin que de très peu de variables internes pour décrire correctement le comportement : la discrétisation de l'éprouvette comporte 210000 degrés de libertés, et seuls 5 modes et la variable d'écrouissage (soit 6 variables internes) suffisent à représenter le modèle homogénéisé. Le gain en temps de calcul n'est pas précisé dans ce cas, mais en général, pour des expériences numériques contenant des millions de degrés de liberté, le gain en temps de calcul et en mémoire est d'au moins 10^6.
Par contre, de même que pour la technique R3M, l'efficacité de la méthode est liée de manière drastique au choix des modes inélastiques. Mais l'inconvénient est la limitation de cette méthode aux petites déformations.

1.2.2 Réduction de modèles dans différents domaines

La méthode de réduction de modèle POD n'est pas seulement vue comme un solveur efficace (comme peut le conclure Ryckelynck dans [Ryckelynck and Missoum-Benziane, 2010]), mais peut être également un outil pour faire de la simulation de chirurgie en temps réel : Niroormandi et al. [Niroormandi et al., 2008] proposent un modèle déformable en temps réel de cornée humaine. La cornée est considérée comme un matériau hyperélastique, donc le modèle employé pour la simulation est un modèle paramétré mécanique hyperélastique. Les paramètres sont les chargements en différentes positions $\underline{\boldsymbol{X}}_k$. Soit $\left[\underline{\boldsymbol{T}}^j(\underline{\boldsymbol{X}}_k)\right]_{k=1,\dots,n_p}$ la réponse du système pour n_p différentes valeurs de paramètres au pas de temps m. L'algorithme 4 décrit la procédure (basée sur une technique POD classique avec interpolation) qui permet d'obtenir la réponse du système en n'importe quelle position du chargement.

BEGIN
1. **for** $k = 1 \rightarrow n_p$ **do**
   Effectuer une simulation complète (Éléments Finis) pour chaque valeur de paramètre;
end
2. Appliquer la POD classique à tous les snapshots du système complet afin d'obtenir une base orthonormée $[\Psi] = [\psi_1, ..., \psi_s]$;
3. **for** $k = 1 \rightarrow n_p$ **do**
   Projeter chaque système de snapshots sur cette nouvelle base : $\underline{\boldsymbol{T}}^j(\underline{\boldsymbol{X}}_k) = [\Psi]\, a_k^j$;
end
4.1 Interpoler à partir de a_k^j la nouvelle valeur de a_X^j parmi les valeurs les plus proches;
4.2 Calculer la nouvelle réponse du système à la nouvelle position du chargement : $\underline{\boldsymbol{T}}^j(\underline{\boldsymbol{X}}) = [\Psi]\, a_X^j$;
END

Algorithm 4: Proper orthogonal Decomposition Interpolation.

Les auteurs soulignent la nécessité de bien choisir les paramètres de départ pour obtenir une bonne base. Une de leurs perspectives est l'utilisation de réduction de modèle a priori APR, avec enrichissement de la base par des sous-espaces de Krylov, afin d'améliorer la précision des simulations.

Dans le cadre de la mécanique des fluides, Bergmann et Cordier [Bergmann and Cordier, 2008] utilisent la technique POD, qui permet d'évaluer une base, optimale au sens de l'énergie et utilisée par la suite pour construire par projection de Galerkin sur les équations du mouvement, un modèle réduit de la dynamique contrôlée de l'écoulement. L'objectif étant de minimiser le coefficient de traînée moyen d'un cylindre circulaire en régime laminaire, les auteurs ont utilisé une procédure d'optimisation couplant modèle réduit (par POD) et méthode à région de confiance (TRPOD). Cette approche sera développée à la section 1.4.1 qui traite des problèmes d'optimisation.

1.2.3 Conclusion

La base POD formée par les vecteurs propres empiriques $(\psi_k)_{k=1...s}$ est optimale d'un point de vue énergétique. Toutefois, une base POD n'est pas capable de représenter une information qui n'était pas contenue initialement dans la base de donnée utilisée pour la déterminer. Le choix des snapshots est donc très important pour représenter l'information nécessaire pour de futures utilisations de la base réduite POD (pour des problèmes d'optimisation par exemple). C'est pourquoi nous proposons de nous intéresser à une

base construite de manière adaptative, au cours de la simulation. En effet, nous avons pu noter que de nombreux auteurs déplorent le fait d'utiliser une base "figée". Ainsi, la prochaine section présentera les méthodes de réduction adaptatives a priori, c'est à dire qui ne nécessitent pas de connaissance au préalable de réponse du système physique.

1.3 Méthodes a priori

Tandis que la plupart des méthodes de réduction de modèles sont des approches dites "a posteriori" (elles sont basées sur des calculs préliminaires pour construire les fonctions de formes du modèle d'ordre réduit, avant de calculer les variables d'état réduites), certaines méthodes permettent d'éviter ces calculs préliminaires, assez coûteux en temps. Pour ces méthodes appelées méthodes "a priori", il n'est pas nécessaire de connaître au préalable un vecteur initial de fonctions de formes pour définir le modèle d'ordre réduit à adapter.
D'autre part, le principal inconvénient de la POD classique et de la snapshot POD est qu'il n'existe pas de procédure adaptative afin de pouvoir modifier la base réduite, selon un indicateur d'erreur. Une première procédure adaptative a été proposée en [Ryckelynck, 2002] en utilisant la méthode LATIN. Mais le caractère non-incrémental du schéma utilisé dans la méthode LATIN pour construire des problèmes linéaires ne facilite pas l'extension de la méthode à n'importe quelle loi de comportement mécanique.

1.3.1 Applications de la méthode A Priori Reduction (APR)

Les algorithmes gloutons décrits en section 1.1.3.2 peuvent aussi être considérés comme des méthodes de réduction a priori non incrémentales dans le cadre de la résolution de problèmes en temps. Dans l'article de Nguyen [Nguyen, 2007], l'auteur propose un algorithme glouton non-incrémental, mais inclut également une notion d'adaptation. En effet, l'objectif des travaux de Nguyen est de construire une base POD robuste, capable de représenter tout un ensemble de dynamiques. Or, puisqu'une base POD est uniquement capable de donner une représentation optimale de l'énergie incluse dans la base de données, cette même base n'est pas adaptée pour représenter une dynamique d'écoulement engendrée avec d'autres paramètres d'entrée.
Son approche est donc basée sur une technique qui consiste à enrichir de façon itérative la base de donnée avec des réalisations calculées là où la borne

de l'erreur commise par le modèle réduit est la plus élevée : il va estimer une erreur a posteriori, et se servir de cet estimateur d'erreur pour calculer la borne de la véritable erreur. Cette borne $\Delta_N(\mu)$ remplace la véritable erreur pour explorer l'espace des paramètres en quête d'une liste de paramètres optimale. Cette approche, nommée la méthode "Greedy Adaptive Sampling", est décrite dans l'algorithme 5.

Data: ;
-Réalisations $u(\mu)$;
-Vecteur de paramètres initiaux $S_N = \{\mu_1, ..., \mu_N\}$;
-Espaces réduits associés $W_N = span\{\xi_n = u(\mu_n)\}_{n=1...N}$;
-Borne de l'erreur : $\Delta_N(\mu)$;
BEGIN
$N = 1$
while $N \leq N_{max}$ **do**
1. Calcul de $\mu_{N+1} = \max_{\mu \in \mathcal{P} \subset \mathcal{D}} \Delta_N(\mu)$;
2. Calcul de $S_{N+1} = \{\{S_N\}, \mu_{N+1}\}$;
3. Calcul du nouvel espace $W_{N+1} = W_N + span\{u(\mu_{N+1})\}$;
4. N=N+1;

end
END

Algorithm 5: Algorithme "Greedy Adaptive Sampling".

La POD sera appliquée ensuite sur les réalisations :

$$S^u_{N_{max}} = \{u(\mu_1), ..., u(\mu_{N_{max}})\}$$

pour obtenir la base POD :

$$\{\xi^{POD}_n\}_{n=1,...,N_{max}}$$

et le nouvel espace d'approximation POD :

$$W^{u,POD}_N = span\{\xi^{POD}_1, ..., \xi^{POD}_N\}_{N=1...N_{max}}$$

.

Les résultats numériques de l'application de cette méthode à un problème de conduction de la chaleur dans un domaine paramétré indiquent que l'approche permet une convergence rapide, mais l'auteur conclut en se posant principalement une question : à savoir si il peut rendre la borne de l'erreur plus rigoureuse, car les estimateurs d'erreurs sont rigoureux seulement dans

certaines situations restreintes. La qualité du modèle réduit n'est donc pas parfaitement maîtrisée.

Ainsi, une estimation d'erreur a posteriori, et l'adaptation de la base ont été proposés pour la réduction d'équations aux dérivées partielles linéaires elliptiques à paramètres non-affines. C'est une approche intéressante puisque les suites de problèmes non-linéaires impliquent en général des suites de problèmes linéaires paramétrés. Mais ce n'est pas facile d'étendre cette méthode aux problèmes mécaniques à variables internes. Dans la plupart des cas, le temps n'est pas un paramètre simple dans les équations aux dérivées partielles. Ces difficultés nous ont amené à adopter une approche incrémentale, et nous allons nous intéresser à l'application de la stratégie de réduction adaptative incrémentale APR "A Priori Reduction". Cette méthode proposée par Ryckelynck [Ryckelynck, 2005] est une méthode a priori itérative, donc la première étape de la méthode est l'initialisation de la base (par un premier calcul complet Éléments Finis sur le premier incrément de temps). Le choix de cette base n'est pas très important, qualitativement parlant. Par contre, mieux elle sera initialisée, plus la convergence de l'algorithme sera rapide. Cette méthode est incrémentale adaptative, et l'adaptation de la base s'effectuera en deux étapes : une sélection des évènements les plus significatifs de la base, suivi de l'enrichissement de celle-ci. Nous n'avons cité là que les principales étapes, mais le détail de la méthode se trouve en section 1.3.2. Nous allons à présent présenter deux principales applications de cette méthode APR.

L'algorithme proposé par Markovinovic et Jansen [Markovinovic and Jansen, 2006] et présenté ci-dessous (Algorithme 6) est assez similaire à celui proposé par Ryckelynck [Ryckelynck, 2005]. Les différences entre les deux méthodes seront exposées juste après.

Data: m=nombre de vecteurs solutions (snapshots);
α=cut-off factor ($0 < \alpha \leq 1$);
BEGIN
$k = 0$
while $k < m$ **do**
1. Résoudre le problème en grande dimension avec une méthode itérative au choix, en utilisant la prévision en base réduite comme solution initiale de la méthode.;
2. $k = k + 1$.;
end
while $k \leq$ *critère d'arrêt (=nombre total de pas de temps)* **do**
3. Former la matrice des snapshots des solutions en grande dimension $[X] := [\underline{\boldsymbol{x}}_{k-m+1}, ..., \underline{\boldsymbol{x}}_k]$.;
4. Décomposer $[X]$ en utilisant la SVD $[X] = [\Theta]\,[\Sigma]\,[\Psi]^T$.;
5. Déterminer $l = \max(\kappa)$ pour lequel $\dfrac{\sum_{j=1}^{\kappa} \sigma_j^2}{\sum_{j=1}^{m} \sigma_j^2} \leq \alpha$.;
6. Déterminer $[\Phi] = [\Theta]\,(:, 1:l)$ i.e utiliser les l premières colonnes de $[\Phi]$ comme fonctions de bases.;
7. Projeter la solution $\underline{\boldsymbol{x}}$ et le système de grande taille sur le sous-espace défini par $[\Phi]$ pour former le système réduit.;
8. Résoudre le système réduit en $\underline{\boldsymbol{z}}$.;
9. Utiliser l'approximation de la solution $\underline{\boldsymbol{x}} \approx [\Phi]\,\underline{\boldsymbol{z}}$ comme solution initiale du solveur itératif poour résoudre le problème de grande taille.;
10. $k = k + 1$.;
end
END

Algorithm 6: Algorithme de POD adaptative proposé par Markovinovic et Jansen.

Il s'agit donc d'une méthode APR comme proposé en [Ryckelynck, 2005]. Des différences mineures peuvent être notées comme le fait que Markovinovic effectue la SVD sur $[X]$ (étape 4) alors que Ryckelynck applique la POD aux variables d'états réduites. Une autre différence à l'étape 1 est l'utilisation d'un solveur itératif (Preconditioned Conjugate Gradient, Minimal Residual ou encore Successive Overrelaxation), alors que Ryckelynck, en 2005, utilisait des sous-espaces de Krylov, et en 2010, une méthode itérative de Newton-

Raphson.

Cet algorithme a été appliqué aux modèles numériques pour la simulation d'un écoulement diphasique en milieu hétérogène poreux, pour lequel seules 10 à 15 fonctions seulement sont nécessaires pour représenter 93500 variables. Les résultats numériques sont concluants, puisque l'efficacité a été prouvée pour les differents solveurs itératifs utilisés, et l'utilisation de cet algorithme permet jusqu'à 67% en gain de temps CPU.

Dans leurs articles [Verdon et al., 2009], Verdon et al. présentent également la méthode adaptative APR, mais pour une autre application : la résolution des équations de transfert du type :

$$\frac{\partial \underline{\boldsymbol{\kappa}}}{\partial t} + \mathcal{F}(\underline{\boldsymbol{\kappa}}) = \mathcal{G} \quad sur \quad \Omega \tag{1.103}$$

$$\underline{\boldsymbol{\kappa}}(.,t) = BC(t) \quad sur \quad \partial\Omega \tag{1.104}$$

$$\underline{\boldsymbol{\kappa}}(\underline{\boldsymbol{x}},t=0) = \underline{\boldsymbol{\kappa}}^{0}(\underline{\boldsymbol{x}}) \tag{1.105}$$

Et ils cherchent à déterminer une base spatiale $\underline{\boldsymbol{\Phi}}_k(\underline{\boldsymbol{x}})$ qui représenterait avec un faible nombre N de vecteurs, la dynamique de l'inconnue $\underline{\boldsymbol{\kappa}}(\underline{\boldsymbol{x}},t)$, telle que :

$$\underline{\boldsymbol{\kappa}}(\underline{\boldsymbol{x}},t) = \sum_{k=1}^{N} \underline{\boldsymbol{\Phi}}_k(\underline{\boldsymbol{x}}) a_k(t) \tag{1.106}$$

La méthode est adaptative, puisqu'à chaque itération de calcul, la base est adaptée : dans un premier temps l'ancienne base est améliorée à l'aide d'une décomposition de Karhunen-Loève tandis que, dans un second temps la base améliorée est enrichie avec des vecteurs de Krylov.
Le dessin présenté en figure FIG. 1.2 et tiré de [Verdon et al., 2009] résume l'algorithme APR.

La méthode a été testée sur le cas simple, entre autres, de l'équation de Burgers 1-D et les résultats ont été comparés à ceux obtenus avec la méthode de Newton-Raphson : la précision de la solution obtenue est très précise mais pas meilleure. Par contre, le temps de calcul est considérablement réduit (30 fois plus rapide pour un maillage comportant 250 noeuds). De plus, les auteurs montrent également la capacité de la base obtenue (contenant seulement 3 modes) à décrire le comportement dynamique de la solution de l'équation de Burgers.

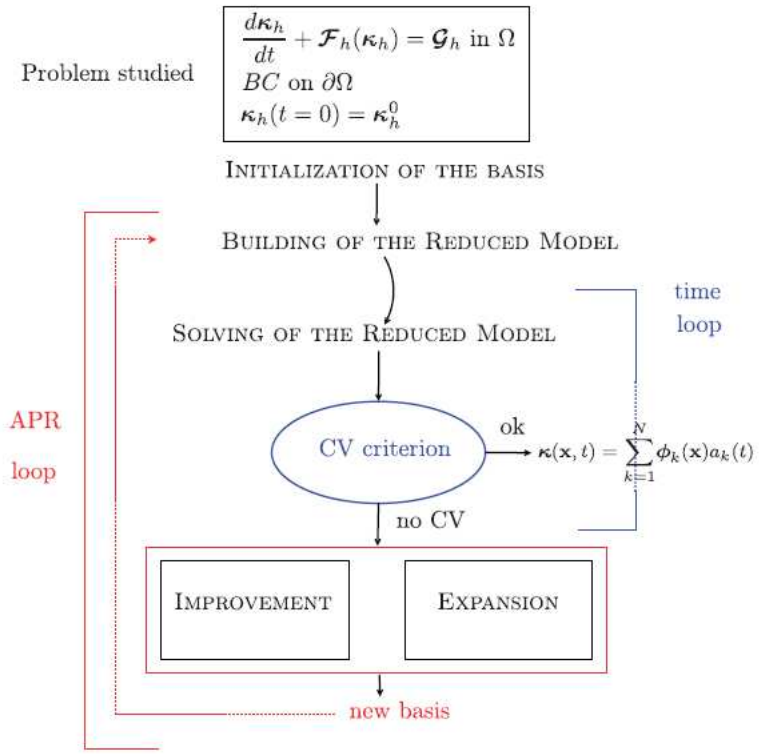


Fig. 1.2 – Algorithme APR [Verdon et al., 2009].

1.3.2 Méthode A Priori Hyper Reduction (APHR)

La méthode APHR a été proposée par Ryckelynck en [Ryckelynck, 2005]. Elle connaît actuellement de nombreux développements dont ce mémoire est une contribution. Dans cette section, nous présentons la méthode telle qu'elle était au début de cette étude doctorale en 2008.
Il s'agit d'une méthode de réduction adaptative de modèle en utilisant un schéma d'intégration en temps classique, incrémental. Un aspect crucial de la méthode est l'introduction de l'Hyper Réduction. La méthode est de ce fait appelée APHR pour A Priori Hyper Reduction (l'Hyper Réduction consistant à ne sélectionner qu'une partie des points d'intégration du modèle Élements Finis pour prévoir l'évolution d'état des variables d'état réduites).
Il est possible de désactiver l'Hyper Réduction dans la méthode APHR. Dans ce cas, on appelle cette méthode "APR".
Ainsi, dans un premier temps, nous présenterons la méthode APR, puis nous rajouterons l'aspect Hyper Reduction.

1.3.2.1 Méthode APR

L'algorithme incrémental implique des périodes adaptatives, pendant lesquelles la simulation incrémentale est recommencée jusqu'à satisfaire un critère de qualité. La procédure adaptative inclut :

- l'extension du sous-espace engendré par les fonctions de forme du modèle d'ordre réduit (en utilisant des sous-espaces de Krylov)
- une sélection des fonctions de formes les plus significatives pour représenter l'évolution d'état.

Les problèmes pour lesquels la méthode est décrite ci-après sont les problèmes thermomécaniques non-linéaires dépendant du temps. Il s'agit de définir les équations du modèle de référence. Dans le cadre de simulations thermomécaniques, la variable d'état $\underline{\boldsymbol{u}}$ peut être un champ de déplacements, ou un champ de températures défini sur toute la structure Ω. La méthode Éléments Finis permet de décrire cette variable d'état à l'aide des fonctions de forme $\underline{\boldsymbol{N}}_i$ et des degrés de libertés $\{q\}_i$ tels que :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t) = \sum_{i=1}^{i=n} \underline{\boldsymbol{N}}_i(\underline{\boldsymbol{X}}\,)\, q_i(t) \quad \forall \underline{\boldsymbol{X}} \in \Omega, \forall t. \tag{1.107}$$

L'évolution d'état est décrite par la valeur des degrés de libertés à différents instants $t_j \in \{t_1, ..., t_m\}$ tel que :

$$q_i(t) = q_i(t_j)\frac{t_{j+1} - t}{t_{j+1} - t_j} + q_i(t_{j+1})\frac{t - t_j}{t_{j+1} - t_j}. \tag{1.108}$$

Soit $\{q\}_j$ un vecteur colonne des variables d'état à l'instant t_j tel que la $i^{\text{ième}}$ composante de $\{q\}_j$ soit $q_i(t_j)$. Supposons qu'un schéma numérique pour l'intégration en temps des équations d'équilibre soit utilisé. Alors, la formulation suivante, pour les conditions d'équilibre, est obtenue :

$$\{q\}_1 = \{q\}_{ini}\,. \tag{1.109}$$

$$\{F\}_{int}\left(\{q\}_{j+1}, .\,\{q\}_j\right) = \{F\}_{ext}\left(\{q\}_{j+1}, .\,\{q\}_j, t_{j+1}\right) \quad \forall j = 1...m-1. \tag{1.110}$$

$\{F\}_{int}$ correspond à la formulation Galerkin des forces internes généralisées, et $\{F\}_{ext}$ correspond au chargement et conditions aux limites appliqués sur la structure Ω. $\{F\}_{int}$, $\{F\}_{ext}$ et $\{q\}$ sont des vecteurs de même taille. Chaque ligne du système de l'équation (1.110) correspond à une condition d'équilibre locale. Une approche pas à pas est utilisée afin d'estimer $\{q\}_{j+1}$ pour $\{q\}_j$ connu.
Soit $(\{q\}_j^{(n)})_{j=1...m}$ la prévision obtenue avec le $n^{\text{ième}}$ modèle d'ordre réduit fourni par la méthode de réduction. Si un algorithme itératif implicite pour résoudre (1.110) est utilisé, seul le dernier résultat obtenu avec cet algorithme itératif est considéré. Celui-ci peut être un algorithme de point fixe, un algorithme de Newton, un algorithme d'Uzawa, ou un algorithme LATIN. Le résidu peut être utilisé pour vérifier que les équations d'équilibre sont bien satisfaites :

$$\{\mathcal{R}\}_{j+1}\left(\{q\}_{j+1}^{(n)}\right) = \{F\}_{int}\left(\{q\}_{j+1}^{(n)}, .\,\{q\}_j^{(n)}\right) - \{F\}_{ext}\left(\{q\}_{j+1}^{(n)}, .\,\{q\}_j^{(n)}, t_{j+1}\right). \tag{1.111}$$

L'estimation de l'évolution d'état $\left(\{q\}_j^{(n)}\right)_{j=1...m}$ est satisfaisante lorsque le critère de qualité suivant est vérifié :

$$\|\,\{\mathcal{R}\}_{j+1}\left(\{q\}_{j+1}^{(n)}\right)\| < \epsilon_R \max_{p\leq j}\left(\|\,\{F\}_{ext}\left(\{q\}_{j+1}^{(n)}, .\,\{q\}_j^{(n)}, t_{j+1}\right)\|\right) \quad \forall j = 1, ..., m-1. \tag{1.112}$$

où ϵ_R est un paramètre de la méthode, défini par l'utilisateur. La norme $\|.\|$ est la norme euclidienne telle que $\|\,\{q\}\,\|^2 = \{q\}^T\{q\}$. Si l'on peut définir une matrice tangente $[K]_{j+1}$, alors la condition suivante est satisfaite :

$$\lim_{\epsilon\to 0}\{\mathcal{R}\}_{j+1}\left(\{q\}_{j+1}^{(n)} + \epsilon\,\{\delta q\}\right) = \{\mathcal{R}\}_{j+1}\left(\{q\}_{j+1}^{(n)}\right) + \epsilon\,[K]_{j+1}\,\delta\,\{q\} \quad \forall\,\{\delta q\}\,. \tag{1.113}$$

Les fonctions de formes $\left(\underline{\boldsymbol{\psi}}_k^{(n)}\right)_{k=1\ldots s}$ du $n^{\text{ième}}$ modèle d'ordre réduit sont déduites des fonctions de formes du modèle de référence grâce à une matrice de base réduite $[A]^{(n)}$. La $k^{\text{ième}}$ fonction de forme $\underline{\boldsymbol{\psi}}_k^{(n)}$ est un champ du même type que $\underline{\boldsymbol{u}}$, et elle est définie par la $k^{\text{ième}}$ colonne de $[A]^{(n)}$, tel que :

$$\underline{\boldsymbol{\psi}}_k^{(n)} = \sum_{i=1}^{i=n} \underline{\boldsymbol{N}}_i(\underline{\boldsymbol{X}})\, A_{ik}^{(n)}. \tag{1.114}$$

Les variables $a_k^{(n)}(t)$ sont les variables d'état réduites du problème réduit telles que :

$$q_i^{(n)}(t) = \sum_{k=1}^{k=s} A_{ik}^{(n)} a_k^{(n)}(t), \tag{1.115}$$

ce qui conduit à :

$$\{q\}_j^{(n)} = [A]^{(n)}\, \{a\}_j^{(n)} \quad \forall j. \tag{1.116}$$

Le modèle d'ordre réduit courant étant toujours défini sur l'intervalle de temps entier, il est facile d'estimer la prédiction sur l'intervalle de temps entier, même si le calcul incrémental de l'évolution d'état a été effectué seulement sur l'intervalle de temps $[t_1, t_{j+1}]$. Pour cela, il suffit de choisir :

$$\{a\}_p^{(n)} = \{a\}_{j+1}^{(n)} \quad \forall p > j+1. \tag{1.117}$$

Évidemment, ce n'est pas la meilleure façon d'estimer l'évolution d'état.

La méthode APHR est basée sur une stratégie adaptative. Ryckelynck propose trois différentes stratégies, mais celle qui est retenue est la stratégie 2 présentée en figure FIG. 1.3.

Le critère de qualité est appliqué à une évolution d'état connue. Cette évolution d'état peut être estimée sur plusieurs pas de temps $[t_\alpha, t_{\alpha+\beta}]$ avant de vérifier la qualité. Si la qualité de l'estimation de l'évolution d'état est satisfaisante, le calcul pas à pas des variables d'état se poursuit sans adaptation. Mais si ce n'est pas le cas, t_α est le début d'une période d'adaptation. La stratégie proposée par Ryckelynck est la suivante : Peu importe la valeur de β, les fonctions de formes sont adaptées, la description de l'état est actualisée à l'instant $t_{\alpha-1}$ et le calcul pas à pas recommence à partir de $t = t_\alpha$. Plus β est petit, plus on pourra maîtriser la qualité des fonctions de forme, et donc la qualité du modèle d'ordre réduit.

L'adaptation se fait en deux étapes :

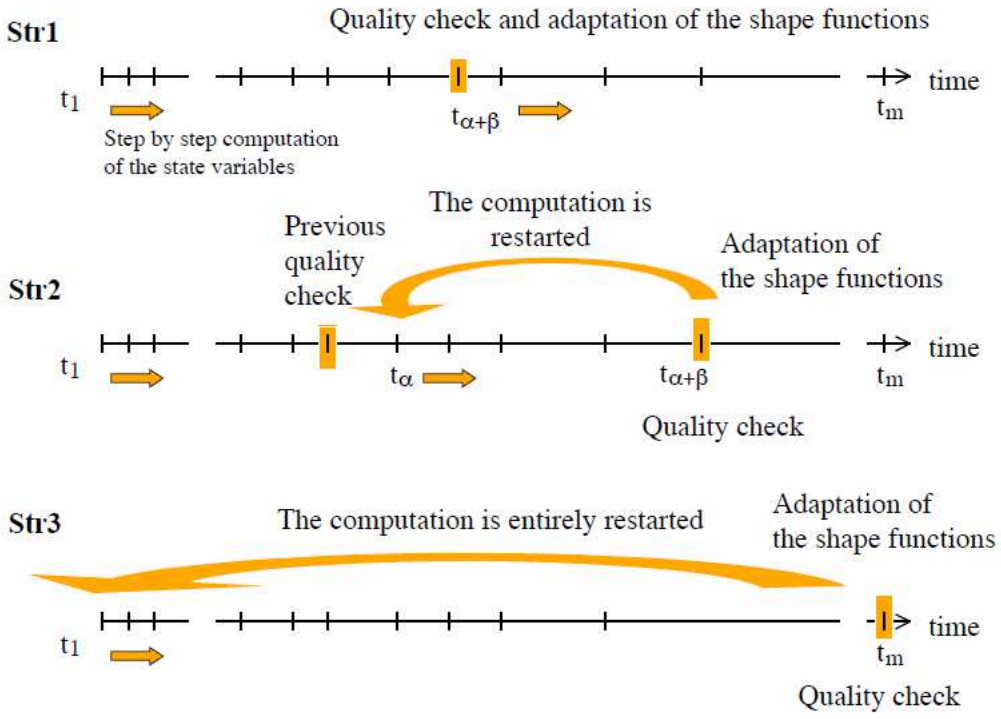


FIG. 1.3 – Stratégies adaptatives, Ryckelynck [Ryckelynck, 2005].

1. Extension du sous-espace par les sous-espaces de Krylov.

Un sous-espace de Krylov est défini par une matrice, un vecteur, et la taille de ce sous-espace. D'habitude, le vecteur et la matrice correspondent respectivement au résidu des équations d'équilibre et à la matrice tangente de rigidité. Soient $\{Y\}_1$ ce vecteur, et $[K]$ cette matrice. Un sous-espace de Krylov est engendré par les vecteurs $(\{Y\}_i)_{i=1...v}$ tel que $\{Y\}_i = [K]\{Y\}_{i-1}$ pour $i > 1$. $\{Y\}_1$ est choisi tel que :

$$\{Y\}_1 = \{\mathcal{R}\}_{j+1}(\{q\}_{j+1}^{(n)}). \tag{1.118}$$

Le vecteur $\{Y\}_{i+1}$ est donné par :

$$\{Y\}_{i+1} = \{\mathcal{R}\}_{j+1}\left(\{q\}_{j+1}^{(n)} + \epsilon_Y \frac{\|\{q\}_{j+1}^{(n)}\|}{\|\{Y\}_i\|}\{Y\}_i\right) - \{\mathcal{R}\}_{j+1}(\{q\}_{j+1}^{(n)}). \tag{1.119}$$

Si la matrice tangente de rigidité existe, et si ϵ_Y est assez petit, alors les vecteurs $(\{Y\}_i)_{i=1...v}$ peuvent engendrer un sous-espace de Krylov.

Les sous-espaces de Krylov sont souvent utilisés pour les méthodes de réduction. Mais aucune d'entre elles n'utilisent directement les vecteurs

$$(\{Y\}_i)_{i=1...v}$$

pour définir les fonctions de forme. Par exemple, dans le cas des méthodes de superposition modale, les modes propres sont calculés par l'algorithme Lanczos [Lanczos, 1952], les fonctions de forme étant les modes propres. Et ces fonctions de forme sont définies grâce aux sous espaces de Krylov. On peut trouver un exemple d'une telle approche appliquée aux problèmes linéaires ou non-linéaires en [Bai, 2002]. Dans le cas des problèmes non-linéaires dépendant du temps, des méthodes de gradient conjugué ont été proposées pour représenter l'évolution d'état par une superposition de fonctions de forme définies par des sous-espaces de Krylov [Risler and Rey, 2000]. Pour ces méthodes également, les fonctions de forme ne correspondent pas directement aux vecteurs définissant les sous-espaces de Krylov. Il faut une procédure d'orthogonalisation pour les construire.
Ryckelynck propose d'utiliser directement les vecteurs $(\{Y\}_i)_{i=1...v}$ comme nouvelles fonctions de forme pour le modèle d'ordre réduit adapté, et ce, même si ces vecteurs ne sont pas réguliers. Dans ce cas, il faudra compter sur l'expansion de Karhunen-Loève pour améliorer la forme de ces fonctions de forme.

2. Sélection des fonctions de forme les plus significatives.

Dans cette étape, la POD est appliquée sur les variables d'état réduites du modèle d'ordre réduit, afin de sélectionner les fonctions de formes correspondant aux évènements principaux contenus dans l'évolution d'état connue. Ainsi, pour la méthode APHR, les variables d'état considérées sont les variables d'état réduites de ce modèle d'ordre réduit, notées $\{a\}^{(n)}$. Le quotient de Rayleigh utilisé est donc :

$$\lambda(\{V\}_k^{(n)}) = \sum_{j=1}^{j=m} \frac{\left(\{a\}_j^{(n)^T} \{V\}_k^{(n)}\right)^2}{\{V\}_k^{(n)^T} \{V\}_k^{(n)}}. \tag{1.120}$$

Et le problème aux valeurs propres est :

$$[H]^{(n)} \{V\}_k^{(n)} = \lambda_k^{(n)} \{V\}_k^{(n)}. \tag{1.121}$$

où $[H]^{(n)}$ est la matrice de covariance suivante :

$$[H]^{(n)} = \sum_{j=1}^{j=m} \{a\}_j^{(n)} \{a\}_j^{(n)^T} \tag{1.122}$$

Les vecteurs propres empiriques appartiennent à l'espace des variables d'état réduites du modèle d'ordre réduit courant. Les s premiers vecteurs propres empiriques définissent une matrice de sélection $[V]^{(n)}$ telle que :

$$[V]^{(n)} = \left[\{V\}_1^{(n)}, \{V\}_2^{(n)}, ..., \{V\}_s^{(n)}\right]. \tag{1.123}$$

Pour obtenir les vecteurs propres empiriques les plus significatifs, s est tel que :

$$\lambda_s^{(n)} \geq \varepsilon_{POD} \lambda_1^{(n)}. \tag{1.124}$$

La matrice de sélection définit un sous-espace de l'espace des variables d'état réduites. Les fonctions de forme correspondant à ce sous-espace sont obtenues par le produit suivant : $[A]^{(n)} [V]^{(n)}$. Grâce à l'expansion de Karhunen-Loève, ces fonctions de formes auraient une meilleure signification physique que les fonctions de forme du modèle d'ordre réduit courant.

Adaptation des fonctions de forme.

Le modèle d'ordre réduit adapté est défini grâce aux nouveaux vecteurs $(\{Y\}_i)_{i=1...v}$ et grâce à la matrice de sélection $[V]^{(n)}$ tel que :

$$[A]^{(n+1)} = \left[[A]^{(n)} [V]^{(n)}, \frac{1}{\| \{Y\}_1 \|} \{Y\}_1, ..., \frac{1}{\| \{Y\}_v \|} \{Y\}_v \right]. \quad (1.125)$$

Les variables intermédiaires sont choisies de sorte que chaque contribution de nouvelles fonctions de formes $\left(\frac{1}{\| \{Y\}_k \|} \{Y\}_k \right)_{k=1...v}$ vaille zéro. De ce fait, on a :

$$\{a\}_j^{(n+\frac{1}{2})} = \begin{bmatrix} [V]^{(n)^T} \{a\}_j^{(n)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (1.126)$$

associés aux nouveaux vecteurs correspondant au sous-espace de Krylov : il y a autant de lignes comportant des 0 que de vecteurs correspondant au sous-espace de Krylov.

Les normes des vecteurs propres empiriques sont telles que les normes des colonnes de la matrice $[A]^{(n+1)}$ restent égales à 1. Les vecteurs $(\{Y\}_i)_{i=1...v}$ seront modifiés par la prochaine matrice de sélection $[V]^{(n+1)}$ de la prochaine adaptation du modèle d'ordre réduit.

1.3.2.2 L'Hyper Réduction

Le but de l'Hyper Réduction est de définir un domaine restreint afin de formuler les équations d'équilibre du modèle d'ordre réduit. En utilisant la projection de Galerkin, les équations d'équilibre peuvent être formulées avec des fonctions tests correspondant aux vecteurs $\{q\}^*$ tel que :

$$\{q\}^* \{F\}_{int} \left(\{q\}_{j+1}, \{q\}_j \right) = \{q\}^{*T} \{F\}_{ext} \left(\{q\}_{j+1}, . \{q\}_j, t_{j+1} \right) \quad \forall \{q\}^* \quad \forall j = 1...m-1. \quad (1.127)$$

Pour le modèle d'ordre réduit courant, Galerkin fournit les équations d'équilibre suivantes :

$$[A]^{(n)^T} \{F\}_{int} \left([A]^{(n)} \{a\}_{j+1}^{(n)}, [A]^{(n)} \{a\}_j^{(n)} \right) = [A]^{(n)^T} \{F\}_{ext} \left([A]^{(n)} \{a\}_{j+1}^{(n)}, [A]^{(n)} \{a\}_j^{(n)}, t_{j+1} \right). \quad (1.128)$$

Ce qui signifie que les résidus $\{\mathcal{R}\}_{j+1} \left([A]^{(n)} \{a\}_{j+1}^{(n)} \right)$ doivent être orthogonaux à l'espace engendré par les fonctions de forme du modèle d'ordre réduit courant. Chaque fonction de forme étant un champ global défini sur toute la structure Ω, une ligne du système (1.128) correspond à une équation d'équilibre globale. Mais il s'agit de satisfaire toutes les équations d'équilibre locales du problème de référence selon le critère de qualité (1.112). Ceci est possible si les fonctions de formes du modèle d'ordre réduit sont satisfaisantes.
Le point clef de cette approche n'est pas l'orthogonalité, mais l'identification de la forme des champs globaux grâce à quelques équations de contrôle. Pour les problèmes non linéaires qui intéressent Ryckelynck, il s'agit de "reconnaître" l'état du système grâce à une combinaison linéaire des fonctions de formes globales. Il choisit donc de n'extraire que quelques équations d'équilibre du problème Éléments Finis pour trouver les équations d'équilibre du modèle d'ordre réduit. Ces équations sont obtenues en introduisant une matrice [Π] remplie de zéros, sauf sur un élément de chaque ligne :

$$[A]^T [\Pi]^T [\Pi] \{F\}_{int} \left([A] \{a\}_{j+1}, [A] \{a\}_j \right) = [A]^T [\Pi]^T [\Pi] \{F\}_{ext} \left([A] \{a\}_{j+1}, [A] \{a\}_j, t_{j+1} \right) \quad \forall j = 1...m-1. \quad (1.129)$$

L'exemple suivant illustre ce que peut être la matrice [Π] pour un problème à 4 degrés de liberté :

$$\begin{Bmatrix} q_1 \\ q_3 \\ q_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{Bmatrix}$$

On introduit ainsi des variables de contrôle, qui sont les degrés de liberté Éléments Finis de noeuds de contrôle. Seuls quelques éléments sont directement connectés à ces noeuds. Soit Ω_Π le domaine d'intégration réduit

(Reduced Integration Domain ou RID) défini par ces éléments. Les autres éléments n'ont donc pas de contribution à la formulation (1.130), donc seuls les points d'intégration de Ω_Π sont utilisés. Ceci permet de réduire le nombre d'équations locales à résoudre. Considérons le domaine de travail Ω ainsi que sa discrétisation représentés dans la figure FIG. 1.4.

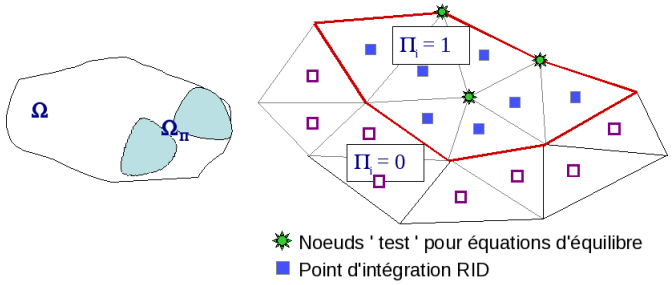


FIG. 1.4 – Définition du domaine réduit d'intégration (RID).

Une fonction test tronquée $\underline{\boldsymbol{u}}^*_\Pi$ est associée à chaque fonction test $\underline{\boldsymbol{u}}^* = \underline{\boldsymbol{N}} \,.\, \{q\}^*$ telle que :

$$\underline{\boldsymbol{u}}^*_\Pi = \underline{\boldsymbol{N}} \,.\, [\Pi]^T \,.\, [\Pi] \,.\, \{q\}^* \,. \tag{1.130}$$

Par conséquent, la propriété suivante est satisfaite :
Si $\underline{\boldsymbol{u}}^* \in \mathcal{V}$ alors $\underline{\boldsymbol{u}}^*_\Pi = \underline{\boldsymbol{N}} \,.\, [\Pi]^T \,.\, [\Pi] \,.\, \{q\}^* \in \mathcal{V}_h$.

En fait, sous réserve de prendre un nombre suffisant d'équations au moins égal ou supérieur au nombre de variables d'état réduites, nous venons de poser une formulation du type Petrov-Galerkin où l'espace des fonctions test est différent de celui des solutions. La différence est dans le fait que cette fonction test vaut zéro sur les degrés de liberté non inclus dans le RID. Cette formulation s'écrit :

$$\int_\Omega \mathcal{L}\left(\underline{\boldsymbol{X}}, t, \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t), \frac{\partial^\alpha \underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^\alpha}, \frac{\partial \underline{\boldsymbol{u}}}{\partial t}\right) \underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}}) d\Omega \;=\; 0, \tag{1.131}$$

$$\forall \underline{\boldsymbol{u}}^* \in \mathcal{V}_\Pi, \forall t \in]0,T], \quad \forall \underline{\boldsymbol{X}} \in \partial_U \Omega \quad \text{avec} \quad \underline{\boldsymbol{u}} \in \mathcal{U}_{POD}$$

où

$$
\begin{aligned}
\mathcal{V}_\Pi \;=\; & \{\, \underline{\boldsymbol{u}} \in \mathcal{V} \,|\exists\, \{a\} \in \mathbb{R}^s \,, \\
& \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha) = \sum_{j=1}^{j=n} \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,)\; q_j(t; \{p\}_\alpha) + \sum_{j=1}^{j=n_c} \underline{\widetilde{\boldsymbol{N}}}_j(\underline{\boldsymbol{X}}\,)\; g_j(t) \\
& \forall \underline{\boldsymbol{X}} \in \Omega \quad \forall t \in]0, T] \\
& \text{avec } \{q\} = [\Pi]^T \;.\; [\Pi] \;.\; [A] \;.\; \{a\} \,\}\,.
\end{aligned} \tag{1.132}
$$

Au final, en exploitant la spécificité de l'espace $\mathcal{V}_\Pi$, la formulation (1.132) devient la condition d'équilibre hyper réduite :

$$
\int_{\Omega_\Pi} \mathcal{L}\left(\underline{\boldsymbol{X}}\,, t, \underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t), \frac{\partial^\alpha \underline{\boldsymbol{u}}}{\partial \underline{\boldsymbol{X}}^{\,\alpha}}, \frac{\partial \underline{\boldsymbol{u}}}{\partial t}\right) \underline{\boldsymbol{u}}^{\,*}(\underline{\boldsymbol{X}}\,) d\Omega \;=\; 0, \tag{1.133}
$$

$$
\forall \underline{\boldsymbol{u}}^{\,*} \in \mathcal{V}_\Pi, \forall t \in]0, T], \quad \forall \underline{\boldsymbol{X}} \in \partial_U \Omega \quad \text{avec} \quad \underline{\boldsymbol{u}} \in \mathcal{U}_{POD}
$$

Avec cette formulation, nous atteignons notre but de réduire le coût des calculs locaux car ces derniers sont restreints au domaine Ω_Π.

Toujours sous la condition d'un RID suffisamment grand pour garantir que le problème mathématique soit bien posé, cette approche permet d'estimer à moindre coût les variables d'état réduites $\{a\}$. Une fois ces variables réduites calculées, le champ de déplacement est obtenu par l'approximation réduite (1.49).

Après chaque adaptation du modèle d'ordre réduit, la liste des noeuds de contrôle est actualisée :

- D'abord, la norme moyenne locale du gradient de chaque fonction de forme est calculée sur chaque noeud du maillage.
- Puis, pour chaque fonction de forme, sont ajoutés à la liste des noeuds de contrôle, ceux dont le maximum de la norme du gradient est atteint.
- Ainsi, grâce à cette procédure, toutes les fonctions de forme seront observables avec les variables de contrôle.

Dans [Ryckelynck, 2005], Ryckelynck propose une méthode de construction de Ω_Π en utilisant les gradients des $(\underline{\boldsymbol{\psi}}_k)_{,k=1,\ldots,s}$ pour des problèmes de thermique. La partie Hyper Réduction de la méthode APHR sera plus amplement détaillée pour les problèmes de mécanique à variables internes en section 2.1.3, et la construction de Ω_Π pour ces mêmes problèmes sera présentée en section 2.4.

1.3.2.3 Conclusion

La méthode APHR multi-niveau fournit des estimations d'états en ajoutant des corrections Éléments Finis aux prévisions réduites. Les variables d'état réduites du modèle d'ordre réduit étant globales, et les variables Éléments Finis étant des inconnues nodales, l'approche proposée est donc multi-niveau. Si, à la fin d'un incrément de temps, la prévision réduite est assez précise, il n'y aura pas besoin de correction Éléments Finis. Dans le cas contraire, les résultats de la correction Éléments Finis permettent d'adapter le modèle d'ordre réduit.
Dès que la correction Éléments Finis est faite, une approximation de l'évolution d'état est connue. Ainsi, on peut appliquer l'algorithme adaptatif proposé en [Ryckelynck et al., 2006], dédié aux évolutions d'états connues. L'état mécanique est pris en compte pour étendre le sous-espace engendré par les bases réduites du modèle d'ordre réduit. Pour maîtriser la croissance du modèle d'ordre réduit, la procédure adaptative nécessite une POD sur les variables d'état réduites.
En conséquence de la méthode d'Hyper Réduction [Ryckelynck, 2009], un schéma d'intégration spatial spécifique est introduit, lors du calcul des variables d'états réduites relatives à la prévision du modèle d'ordre réduit. Les équations constitutives sont intégrées sur un domaine d'intégration réduit, à condition que ces équations soient locales.
La qualité de la prévision du modèle d'ordre réduit est évaluée grâce à la norme du résidu tronqué des équations d'équilibres Éléments Finis. Le calcul de la correction Éléments Finis est effectué en utilisant un algorithme classique incrémental. Celui-ci peut être un algorithme classique de Newton-Raphson.

1.4 Utilisation des modèles d'approximation pour l'optimisation

L'approche générale de l'utilisation des modèles d'approximation pour l'optimisation est d'approcher les équations par un modèle réduit par POD, puis d'appliquer la procédure d'optimisation pour le système réduit. Mais le principal inconvénient est qu'il n'y a aucune assurance mathématique que la solution du problème d'optimisation avec modèle réduit fournira un optimum local pour le problème d'origine. C'est pour cela que des techniques itératives sont nécessaires.
Alexandrov et ses collègues [Alexandrov et al., 1998] décrivent dans un schéma que nous reprenons ici (FIG. 1.5) l'utilisation des modèles d'approximation

en optimisation : de temps en temps, des informations provenant du modèle complet ("High Fidelity") sont utilisées pour vérifier la qualité obtenue avec le modèle d'approximation. Puis, il y a recours au modèle complet pour "recalibrer" le modèle d'approximation, et continuer l'optimisation avec ce modèle simplifié.

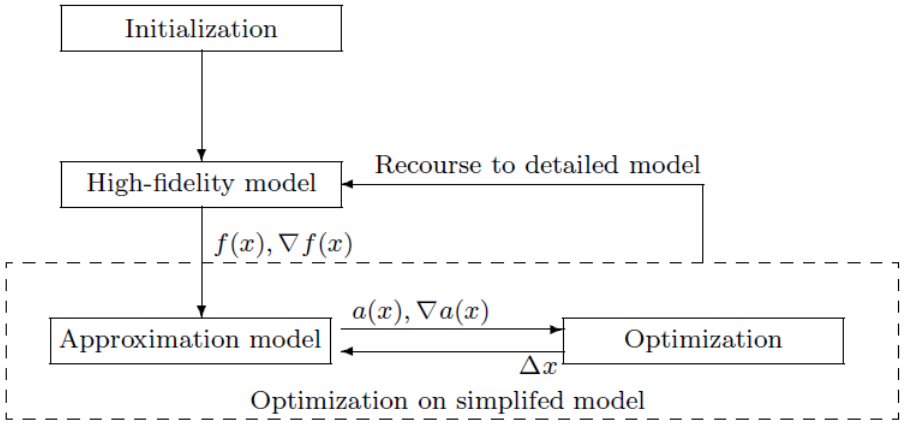


FIG. 1.5 – Algorithme d'optimisation utilisant des modèles d'approximation [Alexandrov et al., 1998].

Alexandrov et al. ont proposé d'utiliser des méthodes à région de confiance. Celles-ci permettent de fournir un critère pour améliorer le modèle réduit lorsque les prévisions sont mauvaises. Dans la section suivante, nous proposons de décrire ces méthodes à région de confiance couplées à la technique POD.

1.4.1 Approche à région de confiance

Malgré l'optimalité énergétique de la base obtenue par POD, il est difficile de construire, une bonne fois pour toute en début du processus d'optimisation, une base POD capable de décrire correctement l'ensemble des réponses fournies par le système au cours du processus d'optimisation. C'est pour cela que la base réduite doit être adaptée pendant le processus d'optimisation. De plus, la réduction de modèle consiste à trouver un système plus simple ayant, dans une certaine gamme de paramètres donnée, le même comportement que le système complet détaillé. Il faut donc déterminer les bonnes gammes et

échelles des paramètres du système, en trouvant, par exemple, une région de confiance pour le modèle réduit. Une méthode efficace consiste à actualiser de facon itérative le modèle pendant le processus d'optimisation. La méthode TRPOD (Trust Region POD) couplant méthode à région de confiance à la POD, initialement introduite par Fahl [Fahl, 2000] est présentée dans cette section, pour une application en mécanique des fluides.
Dans leur étude, Bergmann et Cordier [Bergmann and Cordier, 2008] cherchent à contrôler un écoulement. Ce problème se ramène à minimiser une fonction objectif $\mathcal{J}$ (dans ce cas, le coefficient de traînée moyen) en fonction de n paramètres $\mathbf{c}$ (les variables de contrôle).
Soit ξ_{NS} les variables d'état obtenues par résolution numérique des équations de Navier-Stokes du système. Le problème d'optimisation à résoudre est donc :

$$\min_{\mathbf{c}\in\mathbb{R}^n} \mathcal{J}\left(\xi_{NS}(\mathbf{c}), \mathbf{c}\right) \tag{1.134}$$

Les variables d'état ξ_{NS} peuvent être reconstruites par résolution d'un système réduit en utilisant une base POD. Ces variables sont notées ξ_{POD}, et le problème (1.134) peut être remplacé par :

$$\min_{\mathbf{c}\in\mathbb{R}^n} \mathcal{J}\left(\xi_{POD}(\mathbf{c}), \mathbf{c}\right) \tag{1.135}$$

Le problème est que le modèle réduit construit ne représente la dynamique de l'écoulement que sur une certaine région limitée de l'espace des paramètre de contrôle. C'est cet espace que l'on appelle "région de confiance" de rayon $\Delta^{(k)}$ à l'itéré k de l'algorithme TRPOD que nous allons présenter

dans l'algorithme 7, puis de manière schématique (FIG. 1.6).

Data: ;
$\mathbf{c}^{(0)}$, $\mathcal{J}^{(0)}$, $k = 0$;
BEGIN
while $k < K$ **do**
1. Résolution des équations de Navier-Stokes $\Rightarrow \xi_{NS}(\mathbf{c}^{(k)})$;
2. Extraction des snapshots de l'écoulement : base POD $\underline{\boldsymbol{\psi}}_{i\ i=1,\ldots,N_{POD}}^{(k)}$;
3. Assemblage du modèle réduit MR ;
4. Intégration temporelle du MR pour construire les champs $\xi_{POD}(\mathbf{c}^{(k)})$;
5. Évaluation de la fonction $\mathcal{J}\left(\xi_{POD}(\mathbf{c}^{(k)}), \mathbf{c}^{(k)}\right)$;
6. $m^{(k)}(\mathbf{c}^{(k)} + \mathbf{s}^{(k)}) = \hat{\mathcal{J}}\left(\xi_{POD}(\mathbf{c}^{(k)} + \mathbf{s}^{(k)}), \mathbf{c}^{(k)} + \mathbf{s}^{(k)}\right)$, fonction modèle de la fonction objectif : $f(\mathbf{c}^{(k+1)}) = \mathcal{J}\left(\xi_{NS}(\mathbf{c}^{(k)} + \mathbf{s}^{(k)}), \mathbf{c}^{(k)} + \mathbf{s}^{(k)}\right)$ à l'intérieur de la région de confiance $||\mathbf{s}^{(k)} \leq \Delta^{(k)}||$;
7. Résolution du problème d'optimisation $\min_{\mathbf{s}\in\mathbb{R}^n, ||\mathbf{s}\leq\Delta^{(k)}||} m^{(k)}(\mathbf{c}^{(k)} + \mathbf{s})$;
8. Évaluation de la performance des nouveaux paramètres de contrôle $\mathbf{c}^{(k+1)} = \mathbf{c}^{(k)} + \mathbf{s}^{(k)}$: comparaison de la décroissance réelle *ared* à la décroissance prédite *pred*, avec $ared = f(\mathbf{c}^{(k+1)}) - f(\mathbf{c}^{(k)})$ et $pred = m^{(k)}(\mathbf{c}^{(k+1)}) - m^{(k)}(\mathbf{c}^{(k)})$;
9. **if** *ared est jugée bonne* **then**
le pas $\mathbf{c}^{(k+1)} = \mathbf{c}^{(k)} + \mathbf{s}^{(k)}$ est accepté, et le rayon est augmenté à l'itération suivante : $\Delta^{(k+1)} > \Delta^{(k)}$; $k = k + 1$;
end
if *ared est jugée moyenne* **then**
le pas $\mathbf{c}^{(k+1)} = \mathbf{c}^{(k)} + \mathbf{s}^{(k)}$ est accepté, et le rayon est maintenu à l'itération suivante : $\Delta^{(k+1)} = \Delta^{(k)}$; $k = k + 1$;
end
if *ared est jugée mauvaise* **then**
le pas $\mathbf{c}^{(k+1)} = \mathbf{c}^{(k)} + \mathbf{s}^{(k)}$ est refusé, et le rayon est diminué à l'itération suivante : $\Delta^{(k+1)} < \Delta^{(k)}$; $k = k + 1$; GOTO 7.;
end
end
END

Algorithm 7: Algorithme TRPOD.

L'utilisation de cette procédure TRPOD a non seulement permis de réduire le coefficient de trainée moyen de 30%, mais les coûts de calculs ont été

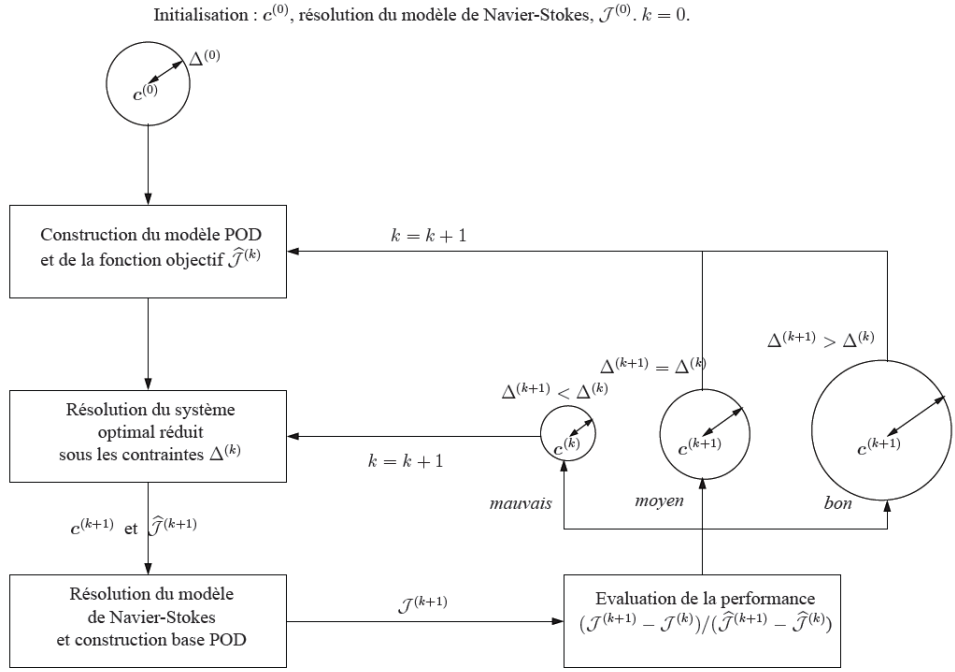


FIG. 1.6 – Schématisation de la méthode à région de confiance [Bergmann and Cordier, 2008].

réduits : 1600 fois moins de valeurs numériques ont été stockées, ce qui a permis une résolution 4 fois plus rapide du système. De plus, la convergence des paramètres de contrôle obtenu du modèle réduit vers ceux, optimaux, du système Navier-Stokes est assurée.

Dans les deux prochaines sections, nous présenterons l'application de méthodes de réduction de modèles à deux types d'approches en optimisation : la résolution de problèmes inverses (correspondant à une manière séquentielle) et la construction de surface de réponse (manière multidimensionnelle).

1.4.2 Problèmes inverses

Le but de cette section est de fournir quelques références sur les problèmes inverses résolus par des méthodes de réduction de modèle.
Un problème inverse est le problème d'optimisation rencontré lors de la comparaison essai/calcul (il faut une approche inverse pour interpréter les mesures).
Le but du problème inverse est d'utiliser les connaissances des données expérimentales relatives à des états non-uniformes, dans le but d'optimiser le modèle du système physique.
En thermique, un des problèmes consiste à estimer les évolutions des sollicitations à partir de températures mesurées (ou obtenues par des simulations numériques) en des points (capteurs) situés au voisinage des surfaces sur lesquelles les sollicitations sont appliquées. La résolution de problèmes inverses consistant à résoudre le modèle détaillé de manière séquentielle à chaque itération, le coût en temps de calcul peut vite devenir important. C'est pour cela que Videcoq et Petit [Videcoq and Petit, 2001], entre autres, proposent l'inversion par un modèle réduit pour identifier plusieurs sollicitations thermiques dans un problème de conduction de la chaleur. Ce modèle réduit est obtenu, avec hypothèse de linéarité, par l'identification de modes propres dominants : le modèle de comportement initial (modèle détaillé) est décomposé sur une base modale, qui est elle même réduite par une simple troncature, afin de ne garder que les modes propres ayant une influence significative sur le problème inverse. La représentation espace-temps du modèle détaillé s'écrit :

$$\left\{ \begin{array}{ll} \{\dot{T}\}(t) & = [A]\{T\}(t) + [B]\{U\}(t) \\ \{Y\}(t) & = [C]\{T\}(t) \end{array} \right. \tag{1.136}$$

où

- $\{T\}(t)$ est le vecteur d'état contenant les N températures nodales,

- $\{U\}(t)$ est le vecteur d'entrée contenant les p densités de flux de chaleur $\varphi(t)$,
- $\{Y\}(t)$ est le vecteur de sortie contenant les températures aux q capteurs,
- $[A](N,N)$, $[B](N,p)$ et $[C](q,N)$ sont respectivement les matrices d'état, d'entrée et de sortie.

Le modèle réduit s'écrit :

$$\begin{cases} \{\dot{X}\}(t) &= [F]\{X\}(t) + [G]\{\dot{U}\}(t) \\ \{\hat{Y}\}(t) &= [H]\{X\}(t) + [S]\{U\}(t) \end{cases} \tag{1.137}$$

où

- $\{X\}(t)$ est le nouveau vecteur d'état réduit dans la base modale de dimension $n < N$,
- $\{\hat{Y}\}(t) \approx \{Y\}(t)$,
- $[F](n,n)$ la matrice diagonale des valeurs propres qui contient les n modes dominants du système,
- $[S](q,p)$ la matrice de sensibilité statique,
- $[G](n,p)$, $[H](q,n)$ respectivement les nouvelles matrices réduites d'entrée et de sortie.

Pour établir le modèle réduit (1.137), plusieurs modèles réduits élémentaires (ERM) sont utilisés : chaque composante $u(t) = U_i(t)_{i=1,\dots,p}$ est associée à un ERM qui s'écrit :

$$\begin{cases} \{\dot{x}\}(t) &= [f]\{x\}(t) + \{g\}\{\dot{u}\}(t) \\ \{\hat{y}\}(t) &= [h]\{x\}(t) + \{s\}\{u\}(t) \end{cases} \tag{1.138}$$

où

- $\{x\}(t)$ de dimension m est relatif à $u(t)$,
- $[f](m,m)$ la matrice diagonale relative au ERM courant,
- le vecteur statique $\{s\}$ est la $i^{\text{ème}}$ colonne de $[S]$.

Les modes dominants ne sont pas calculés en résolvant un problème aux valeurs propres, ils ne sont pas sélectionnés comme dans les méthodes modales que nous avons présentées (comme la POD). Les composantes des matrices, ainsi que l'ordre m sont identifiés par une procédure de minimisation d'un critère quadratique relatif aux différences entre les simulations provenant du modèle détaillé (DM) et les résultats de chaque ERM. Ce critère s'écrit, à un ordre m :

$$J_{red}([f],[h],\{g\}) = \sum_{i=1}^{q}\sum_{k=1}^{n_t}\left[y_{ik}(DM) - y_{ik}(ERM)\right]^2 \tag{1.139}$$

Ce problème d'optimisation est résolu grâce à une méthode des moindres carrés et de gradient conjugué. Cette méthode est donc itérative, et s'arrête à l'ordre m lorsque $J_{red}(m+1) \approx J_{red}(m)$. Les valeurs de $[f], [h], \{g\}$ sont optimales et sont conservées pour l'ERM.
Une fois que les p ERM sont identifiés, grâce au principe de superposition, les matrices du modèle réduit (1.137) sont de la forme :

$$[H] = \left[\begin{array}{ccccccc} [h_1] & \dots & [h] & \dots & [h_p] \end{array} \right], \text{ et } [S] = \left[\begin{array}{ccc} \{s\}_1 & \dots & \{s\} \dots \{s\}_p \end{array} \right]$$

$$\begin{aligned} \{X\} &= \left[\begin{array}{c} \{x\}_1 \\ \vdots \\ \{x\} \\ \vdots \\ \{x\}_p \end{array} \right], [F] = \left[\begin{array}{ccccc} [f_1] & & & & \\ & \ddots & & & \\ & & [f] & & \\ & & & \ddots & \\ & & & & [f_p] \end{array} \right] \\ [G] &= \left[\begin{array}{ccccc} \{g\}_1 & & & & \\ & \ddots & & & \\ & & \{g\} & & \\ & & & \ddots & \\ & & & & \{g\}_p \end{array} \right] \end{aligned} \tag{1.140}$$

et l'ordre du modèle réduit (1.137) est $n = \sum_{i=1}^{p} m$.
Les ERM étant découplés, le processus d'inversion peut être appliqué à un problème de conduction de la chaleur tridimensionnel : un cube en acier inoxydable, sur lequel sont imposées comme conditions aux bords 3 densités de flux de chaleur $\varphi_1(t)$, $\varphi_2(t)$, et $\varphi_3(t)$ formant la matrice $[U]$. Dans leur étude, les auteurs ne considèrent pas des mesures de températures, mais des températures obtenues par simulation grâce au modèle détaillé. Le modèle détaillé est d'ordre $11 \times 11 \times 11 = 1331$.
3 capteurs sont situés au centre des 3 faces opposées, les températures nodales obtenues en ces 3 points sont stockées dans la matrice $[Y]$.
Seuls 3 ERM sont identifiés, donc le modèle réduit correspondant est d'ordre 9 (3 pour chaque ERM).
La qualité des résultats obtenus est tout à fait acceptable.
Ainsi, cette approche est intéressante car elle ne nécessite pas une modélisation détaillée toujours coûteuse en temps de calcul et délicate à mettre en place, mais l'inconvénient est qu'elle n'est applicable qu'à des systèmes linéaires.

Une méthode dérivée de cette méthode d'identification modale développée pour les systèmes linéaires est proposée par Girault [Girault and Petit, 2005a] pour résoudre des problèmes inverses de conduction de la chaleur non-linéaires multidimensionnels [Girault and Petit, 2005b]. En séparant les termes linéaires et non-linéaires du système discret, la représentation espace-temps du modèle détaillé s'écrit :

$$\left\{ \begin{array}{ll} \{\dot{T}\}(t) & = [A]\{T\}(t) + [B]\{U\}(t) + \{\Psi\}(\{T\}(t)) \\ \{Y\}(t) & = [C]\{T\}(t) \end{array} \right. \tag{1.141}$$

où $[A]$ est la matrice d'état d'un système linéaire (cf (1.136)), et $\{\Psi\}(\{T\}(t))$ est le vecteur de dimension N rassemblant les non-linéarités. Avec les mêmes notations qu'en (1.137), la transformation $\{T\}(t) = [M]\{X\}(t)$ injectée dans (1.141) permet d'écrire le modèle réduit de cette manière :

$$\left\{ \begin{array}{ll} \{\dot{X}\}(t) & = [F]\{X\}(t) + [G]\{\dot{U}\}(t) + [\Omega]\{Z\}(\{X\}(t)) \\ \{\hat{Y}\}(t) & = [H]\{X\}(t) \end{array} \right. \tag{1.142}$$

où $[M]$ est la matrice de dimension (N, N) contenant les vecteurs propres de $[A]$.
On peut toujours réécrire $[M]^{-1}\{\Psi\}([M]\{X\}(t)) = [M]^{-1}[L]\{Z\}(\{X\}(t))$. Comme dans [Videcoq and Petit, 2001], la méthode de minimisation quadratique est utilisée pour identifier les composantes des matrices $[F], [H], [\Omega]$ ainsi que l'ordre n. Le critère est le suivant :

$$J_{red}(n, [F], [\Omega], [H]) = \sum_{i=1}^{q} \sum_{k=1}^{n_t} [y_{ik}(DM) - y_{ik}(RM)]^2 \tag{1.143}$$

La différence réside dans la résolution de ce problème d'optimisation : en effet, $\{\hat{Y}\}(t)$ étant non-linéaire par rapport à $[F]$ et $[\Omega]$, c'est une méthode de Quasi-Newton qui est utilisée pour l'identification des composantes de ces matrices. La méthode des moindres carrés est conservée pour l'identification de $[H]$, puisque $\{\hat{Y}\}(t)$ est linéaire par rapport à $[H]$.
Ainsi, de nombreuses résolutions d'équations différentielle non linéaires du premier ordre sont nécessaires, mais le nombre réduit n d'équations a permis aux auteurs de diviser le temps CPU par un facteur de plus de 1000.

L'article de Balima et al. [Balima et al., 2006] présente une comparaison entre ces méthodes d'identification modale et l'utilisation de la POD.

La méthode que nous venons de présenter permet d'identifier les modes dominants avec une méthode d'identification modale, mais nous allons maintenant présenter un exemple d'utilisation de la POD pour construire des modèles réduits utilisés pour résoudre des problèmes inverses non-linéaires de conduction [Park et al., 1999]. Ce problème consiste à estimer la fonction inconnue $G(t)$ représentant une source de chaleur dépendant du temps, à partir des températures mesurées par un thermocouple en un point donné. Le modèle doit être capable de prédire la distribution de température sur le domaine pour une fonction $G(t)$ donnée. La décomposition de Karhunen-Loève appliquée sur un ensemble de 600 snapshots (les champs de températures) permet d'obtenir 600 modes empiriques, mais seuls 10 sont utilisés pour construire le modèle réduit. Le nombre de degrés de liberté du système d'équations à résoudre est donc passé de 1600 (domaine carré $2D$, 40×40 noeuds) à 10.
La minimisation de la fonction objectif est faite par une méthode de gradient conjugué.
Le résultat le plus important présenté par les auteurs est la comparaison des temps CPU : 6s pour la résolution du problème inverse en utilisant le modèle réduit, contre 15min 21s avec le modèle détaillé pour obtenir le même résultat.

D'autres algorithmes LATIN ont été proposés pour résoudre des problèmes d'optimisation pour la résolution de problèmes inverses [Allix and Vidal, 2002].

1.4.3 Construction de surfaces de réponse

Dans le contexte de l'optimisation, une méthode de surface de réponse est une approche du type "entrée-sortie", où, pour un jeu de paramètres donnés, le modèle doit fournir une réponse. Ainsi, pour une série de jeux de paramètres, des résolutions du modèle physique sont effectuées, et, en chacun de ces points, les quantités à optimiser sont calculées. Grâce à cette base de données, la fonction coût est approximée sur l'espace des paramètres d'optimisation, conduisant ainsi à un problème d'optimisation approché. Le schéma représentant la résolution d'un problème d'optimisation par une méthode de surface de réponses (tiré de la thèse de Do [Do, 2006]) peut être trouvé en figure FIG. 1.7.

Nous ne nous intéressons pas aux différentes techniques d'optimisation par surface de réponse, mais à la création de ces surfaces de réponse pour résoudre des problèmes d'optimisation. Il existe des techniques de réduction de modèle utilisées pour construire ces bases de données, notamment une méthode d'interpolation des bases construites par POD.

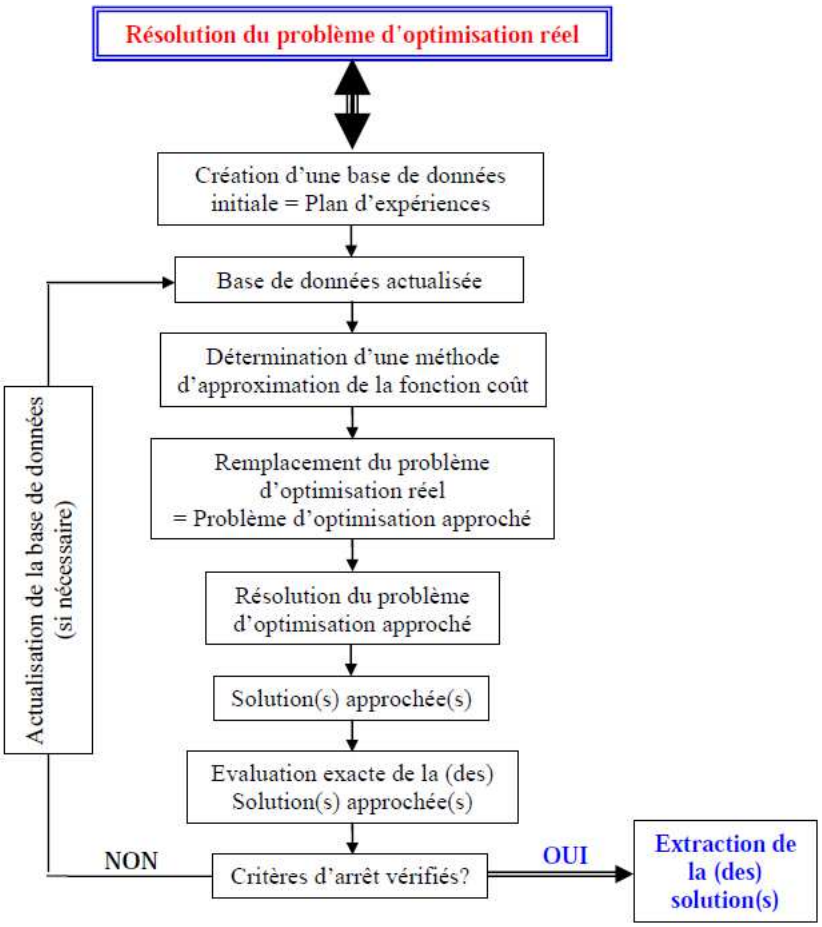


FIG. 1.7 – Résolution d'un problème d'optimisation par une méthode de surface de réponse [Do, 2006].

Lieu et al. [Lieu et al., 2006] étudient un modèle réduit pour écoulement autour d'un F-16. Ils utilisent la POD en mécanique des fluides pour construire des bases réduites, utilisées pour obtenir un modèle d'ordre réduit pour un nombre de Mach d'écoulement libre donné.
Cette base réduite ne permet pas de capter la solution pour des nombres Mach éloignés du point d'équilibre. Il faut donc adapter la base aux variations de paramètres, et cela se fait par la méthode "subspace angle interpolation" proposée dans cet article. Pour cela, il faut connaître les bases POD Φ_1 et Φ_2 construites respectivement pour les nombres Mach M_{∞_1} et M_{∞_2}. Cette méthode est basée sur

- l'interpolation des angles principaux à partir de l'angle principal formé par le sous espace engendré par les bases Φ_1 et Φ_2,
- l'utilisation de l'angle interpolé pour obtenir les vecteurs de bases interpolés, qui forment la nouvelle base POD pour un nombre de Mach intermédiaire $M_{\infty_1} \leq M_\infty \leq M_{\infty_2}$.

Cette méthode est appliquée à l'identification paramétrique aéroélastique d'un F-16. Elle a été testée pour des nombres de Mach compris entre 0.6 et 0.8 avec de bons résultats. Cependant, d'après les auteurs eux-mêmes, cette méthode est limitée, puisque les résultats ne sont valables que pour de faibles écarts entre les nombres de Mach. En effet, comme nous l'avons déjà précisé, un modèle d'ordre réduit est construit pour un jeu de paramètres donné (ici le Mach), et il ne permettra pas de bien représenter la dynamique du fluide pour un Mach différent de celui avec lequel il a été construit. Il faudrait reconstruire un nouveau modèle réduit, mais évidemment, cela a un coût. Par exemple, les auteurs parlent de 3.85 heures CPU sur un cluster pour construire un modèle d'ordre réduit avec 69 degrés de liberté. 95% de ce temps CPU provient du calcul des 99 snapshots associés au modèle Éléments Finis (comportant 168799 degrés de liberté). Par contre, une fois le modèle d'ordre réduit construit, la prévision d'une réponse est de l'ordre de la minute sur le même cluster. Cet exemple illustre bien la différence entre l'exploitation du modèle d'ordre réduit, et le temps mis pour le construire, et il met en évidence le besoin d'un algorithme qui adapterait le modèle d'ordre réduit aux variations de paramètres.
Amsallem et Farhat [Amsallem and Farhat, 2008] proposent une telle méthode d'adaptation qu'ils décrivent en un algorithme à 4 étapes. Cet algorithme nécessite la connaissance de N_R bases réduites construites aux N_R points d'opérations $\lambda_{i,(i=1\ldots N_R)}$, et des N_R sous-espaces $\mathcal{S}_i$ associés. Soit N_p le nombre de paramètres par point d'opération. L'objectif est d'interpoler les bases réduites données afin de construire rapidement un nouveau modèle d'ordre réduit fiable en un nouveau point λ_{N_R+1}.

- Choix d'un point $\mathcal{S}_{i_0}$ de la variété de Grassmann, qui sera la référence

et le point d'origine pour l'interpolation.

- Transfert de la base réduite vers un espace sans contrainte (l'espace tangent $\mathcal{T}_{\mathcal{S}_{i_0}}$ en $\mathcal{S}_{i_0}$). Chaque point $\mathcal{S}_i$ suffisamment proche de $\mathcal{S}_{i_0}$ est représenté par une matrice $[\Gamma_i]$, représentant un point $\mathcal{X}_i$ de $\mathcal{T}_{\mathcal{S}_{i_0}}$ en utilisant la projection logarithmique $Log_{\mathcal{S}_{i_0}}$, comme on peut le voir sur la figure FIG. 1.8.

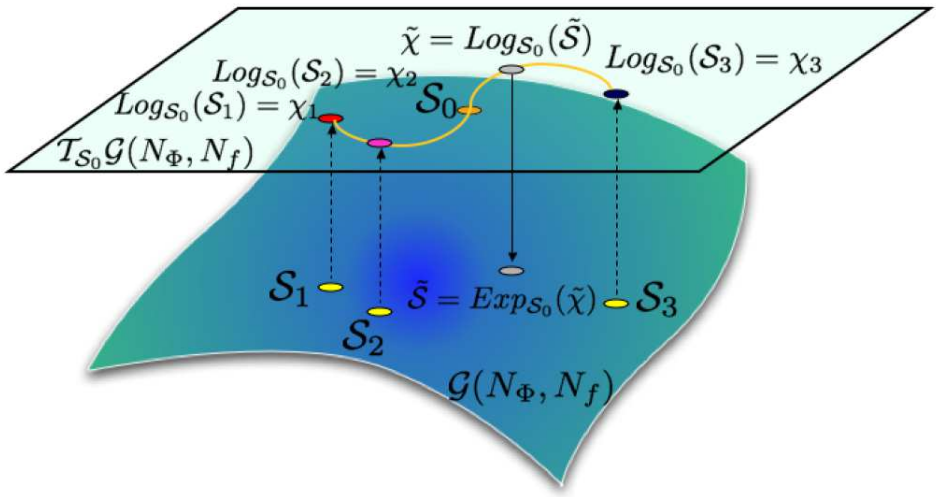


FIG. 1.8 – Description de la projection logarithmique, et de l'interpolation de 4 sous-espaces dans un espace tangent d'une variété de Grassmann [Amsallem and Farhat, 2008].

- Chaque entrée de la matrice $[\Gamma]_{N_R+1}$ associée au point d'opération λ_{N_R+1} est calculée en interpolant, dans cet espace, les entrées correspondantes des matrices $[\Gamma]_i$ associées aux points d'opération λ_i. La méthode d'interpolation est de type Lagrange, par exemple, si $N_p = 1$.
- La matrice $[\Gamma]_{N_R+1}$ représentant $\mathcal{X}_{N_R+1} \in \mathcal{T}_{\mathcal{S}_{i_0}}$ est projetée dans le sous-espace $\mathcal{S}_{N_R+1}$ de la variété de Grassmann, en utilisant la projection exponentielle $Exp_{\mathcal{S}_{i_0}}$.

Les auteurs émettent une nuance sur le terme d'adaptation : la procédure d'adaptation décrite dans l'algorithme ci-dessus est décrite comme une méthode d'interpolation dans un espace tangent à une variété de Grassmann, comme la montre la figure FIG. 1.8. En effet, ce n'est pas vraiment une adaptation, puisque l'adaptation requiert une mesure d'erreur (erreur que l'on essaie de réduire au maximum), alors qu'il n'y a pas de contrôle de l'erreur dans un processus d'interpolation. Toutefois, l'efficacité de cette méthode a été prouvée

par les auteurs pour l'adaptation des modèles d'ordre réduits aéroélastiques à des nouvelles valeurs de Mach, évitant ainsi la reconstruction d'un nouveau modèle d'ordre réduit. Ils obtiennent de bonnes corrélations avec les résultats obtenus en reconstruisant un nouveau modèle d'ordre réduit.

Xiao et al. [Xiao et al., 2009] se sont également intéressés au développement et à l'utilisation de modèles réduits basés sur la POD pour l'optimisation de forme d'un conduit d'admission. La décomposition POD d'un champ $\underline{\boldsymbol{p}} = (p_1, p_2, ..., p_n)^T$, où n est le nombre de noeuds d'un maillage, est investiguée par un plan d'expériences $\{\underline{\boldsymbol{p}}^{(k)}; k = 1, ..., M\}$ et l'approximation POD s'écrit :

$$\underline{\boldsymbol{p}}^{POD,(k)} = \overline{\underline{\boldsymbol{p}}} + \underline{\boldsymbol{\Phi}}_{,m}\underline{\boldsymbol{\alpha}}^{(k)} \tag{1.144}$$

où $\overline{\underline{\boldsymbol{p}}}$ est la contribution moyenne sur les M snapshots, $\underline{\boldsymbol{\Phi}}_{,m}$ la base POD tronquée à m modes, et $\underline{\boldsymbol{\alpha}}^{(k)}$ les coefficients de la combinaison linéaire.
L'erreur de troncation de la base vectorielle obtenue par POD est acceptable en résolution, mais peut être pénalisante dans le cadre d'utilisation d'un tel modèle réduit en optimisation. De plus, elle entraîne inévitablement une non-conservation de grandeurs intégrales, notamment celles qui seraient définies comme des applications linéaires ℓ sur $\underline{\boldsymbol{p}}$:

$$q^{POD,(k)} = \ell(\underline{\boldsymbol{p}}^{POD,(k)}) \neq q^{(k)} = \ell(\underline{\boldsymbol{p}}^{(k)}) \tag{1.145}$$

Pour pallier cette limitation, ils ont développé une méthode basée sur la décomposition aux valeurs propres contraintes (CPOD) en imposant que les coefficients de la décomposition soient tels que des grandeurs scalaires obtenues par des intégrations des vecteurs satisfassent des contraintes d'égalité strictes, au moins pour les points du plan d'expériences.
La méthode consiste à remplacer les coefficients $\underline{\boldsymbol{\alpha}}^{(k)}$ par les coefficients $\underline{\boldsymbol{\gamma}}^{(k)}$ en gardant la même base Φ.
Ces coefficients $\underline{\boldsymbol{\gamma}}^{(k)} = \left(\gamma_1^{(k)}, ..., \gamma_m^{(k)}\right), \quad k = 1, ..., m$ sont déterminés de manière à minimiser l'erreur $J(\underline{\boldsymbol{\gamma}}^{(k)})$ commise par l'approximation, tout en imposant la conservation des r applications linéaires ℓ_j, $j = 1, ..., r$:

$$\begin{cases} J(\underline{\boldsymbol{\gamma}}^{(k)}) & = & \frac{1}{2}\|\underline{\boldsymbol{p}}^{(k)} - \left(\overline{\underline{\boldsymbol{p}}} + \underline{\boldsymbol{\Phi}}_{,m}\underline{\boldsymbol{\gamma}}^{(k)}\right)\|^2 \\ \text{soumis à} & : & \ell_j(\overline{\underline{\boldsymbol{p}}} + \underline{\boldsymbol{\Phi}}_{,m}\underline{\boldsymbol{\gamma}}^{(k)}) = \ell_j(\underline{\boldsymbol{p}}^{(k)}) = \underline{\boldsymbol{c}}_j; \quad j = 1, ..., r \end{cases} \tag{1.146}$$

En définissant le lagrangien, où $\underline{\boldsymbol{W}}$ est la matrice définissant les r applications linéaires

$$\mathcal{L}(\underline{\boldsymbol{\gamma}}^{(k)}, \underline{\boldsymbol{\lambda}}) = J(\underline{\boldsymbol{\gamma}}^{(k)}) + \underline{\boldsymbol{\lambda}}.\left(\underline{\boldsymbol{W}}^T.(\overline{\underline{\boldsymbol{p}}} + \underline{\boldsymbol{\Phi}}_{,m}\underline{\boldsymbol{\gamma}}^{(k)}) - \underline{\boldsymbol{c}}\right) \tag{1.147}$$

et en écrivant les conditions d'optimalité du premier ordre par rapport à $\underline{\boldsymbol{\gamma}}^{(k)}$ et $\underline{\boldsymbol{\lambda}}$, ils aboutissent au système sous forme matricielle suivant, où $\underline{\underline{\boldsymbol{M}}}$ est la matrice de masse :

$$\begin{bmatrix} \underline{\boldsymbol{\Phi}}_{,m}^{T} \underline{\underline{\boldsymbol{M}}} \underline{\boldsymbol{\Phi}}_{,m} & \underline{\boldsymbol{\Phi}}_{,m}^{T} \underline{\boldsymbol{W}} \\ \underline{\boldsymbol{W}}^{T} \underline{\boldsymbol{\Phi}}_{,m} & \underline{\mathbf{0}} \end{bmatrix} \begin{Bmatrix} \gamma^{(k)} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} \underline{\boldsymbol{\Phi}}_{,m}^{T} \underline{\underline{\boldsymbol{M}}} (\underline{\boldsymbol{p}}^{(k)} - \overline{\underline{\boldsymbol{p}}}) \\ \underline{\boldsymbol{c}} - \underline{\boldsymbol{W}}^{T} \overline{\underline{\boldsymbol{p}}} \end{Bmatrix} \quad (1.148)$$

Cette nouvelle méthode est combinée avec une méthode d'approximation par krigeage des coefficients. La stratégie complète se résume en quatre étapes :

- 1. exécution d'un plan d'expériences de simulations numériques "haute fidélité" pour obtenir M snapshots
- 2. calcul de la contribution moyenne $\overline{\underline{\boldsymbol{p}}}$ et de la base POD $\underline{\underline{\boldsymbol{\Phi}}}$
- 3. calcul des coefficients $\gamma^{(k)}$ par CPOD (eq. 1.148)
- 4. à partir des données calculées aux points 2. et 3., un métamodèle généraliste permet de déterminer pour n'importe quel jeu de variables de conception $\mathbf{v}$ une approximation $\tilde{\gamma}^{(k)}(\mathbf{v})$ des coefficients CPOD correspondants, et donc de reconstruire le champ $\underline{\boldsymbol{p}}$, et d'en déduire les grandeurs post traitées. Dans le travail des auteurs, c'est un métamodèle basé sur le krigeage qui est utilisé.

Les résultats obtenus ont permis de constater que les valeurs obtenues pour les points du plan d'expérience coïncident exactement avec les points initiaux calculés à partir de la simulation haute fidélité.

Ces différentes approches proposées sont des approches séquentielles de construction de bases de données pour les surfaces de réponses. Dans le chapitre 4, nous construirons ces bases de données de manière simultanée.

Chapitre 2

Méthode APHR en mécanique non linéaire.

L'avancée récente dans la méthode d'Hyper Reduction proposée en [Ryckelynck, 2009] permet d'étendre la méthode APHR aux modèles mécaniques non-linéaires à variables internes. Mais l'extension du sous-espace par les sous-espace de Krylov n'a pas été conservée. En effet, cette approche augmente le nombre d'évaluations du résidu d'équilibre. Ainsi, le temps de calcul consacré au calcul du résidu devient beaucoup trop coûteux dans le cas des lois de comportement complexes. Nous proposons donc d'étendre la base réduite en utilisant la solution des équations Éléments Finis sur quelques incréments de temps. Ces corrections globales non-linéaires seront détaillées dans la section 2.1.1. Elles introduisent une approximation multi-niveau. Le premier niveau est l'approximation réduite, le second niveau est la correction Éléments Finis. Cette méthode de réduction adaptative est appelée méthode APHR multi-niveau. La formulation de la méthode APHR proposée par Ryckelynck [Ryckelynck, 2005] en thermique et détaillée en section 1.3.2 est reformulée dans les sections suivantes pour les modèles mécaniques non-linéaires à variables internes [Ryckelynck and Missoum-Benziane, 2010]. La stratégie d'adaptation utilisée est celle proposée par Ryckelynck (voir FIG. 1.3) avec $\beta = 1$.

2.1 Formulation du problème mécanique

La fonction inconnue $\underline{\boldsymbol{u}}$ que l'on cherche à trouver dans la formulation du problème (1.1) est un champ de déplacements, et l'opérateur $\mathcal{L}$ utilisé est un opérateur qui agit sur ce champ de déplacements, et sur ses dérivées

partielles. En mécanique, et en transformations finies,

$$\mathcal{L} = \text{div}\underset{\sim}{\boldsymbol{S}} \tag{2.1}$$

où $\underset{\sim}{\boldsymbol{S}}$ est le premier tenseur de contraintes de Piola-Kirchhoff.
Pour faciliter la formulation de l'algorithme, on se limite au cas des hypothèses de petites déformations, petits déplacements, pour lesquelles la méthode des Éléments Finis permet d'obtenir une solution approchée. La formulation en transformation finies sera donnée au chapitre 4.

$$\mathcal{L} = \text{div}\underset{\sim}{\boldsymbol{\sigma}} \tag{2.2}$$

où $\underset{\sim}{\boldsymbol{\sigma}}$ est la contrainte de Cauchy (fonction non linéaire de la déformation) reliée, pour des déformations infinitésimales, à l'histoire du tenseur de déformation $\underset{\sim}{\boldsymbol{\varepsilon}}_\tau$ tel que :

$$\underset{\sim}{\boldsymbol{\sigma}} = \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau\ ,\ \tau \leq t; \{p\}_\alpha) \tag{2.3}$$

où $\underset{\sim}{\boldsymbol{\Sigma}}$ est un opérateur formel qui définit la relation de comportement, et $\underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}) = \frac{1}{2}\left(\underset{\sim}{\boldsymbol{G}}rad(\underline{\boldsymbol{u}}) + \underset{\sim}{\boldsymbol{G}}rad^T(\underline{\boldsymbol{u}})\right)$. Nous considérons que les paramètres $\{p\}_\alpha$ sont ceux du modèle de la relation de comportement. Les relations de comportement sont supposées locales en espace, l'opérateur $\underset{\sim}{\boldsymbol{\Sigma}}$ ne fait intervenir que l'histoire de $\underset{\sim}{\boldsymbol{\varepsilon}}_\tau$ au point matériel considéré.

2.1.1 Formulation du problème mécanique de référence

2.1.1.1 Formulation du modèle continu

Considérons un problème mécanique classique dans le cadre de l'hypothèse des petites perturbations (petits déplacements et petites déformations). Les équations régissant le comportement sont considérés comme non linéaires. Le modèle continu est un modèle mécanique paramétré (on rappelle que $\{p\}$ est le vecteur colonne contenant les paramètres du modèle).
Il n'est pas nécessaire de donner le détail des équations de la relation de comportement pour décrire la méthode proposée.

Le problème mécanique s'énonce comme suit :

Pour une valeur donnée du vecteur de paramètres, le champ de déplacements recherché doit être cinématiquement admissible et les champs de déformations et de contraintes qui lui sont associés doivent vérifier les conditions d'équilibre et les équations de la relation de comportement. On veut donc trouver une estimation du champ de déplacement $\underline{\boldsymbol{u}} \in \mathcal{U}$ *satisfaisant les équations de*

comportement du milieu continu et le principe des travaux virtuels. Dans le cas de problèmes mécaniques, (1.6) revient à :

$$\int_\Omega \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) \,:\, \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}),\, \tau \leq t; \{p\}_\alpha)\; d\Omega \;-\; \int_{\partial_f\Omega} \underline{\boldsymbol{u}}^* \,.\, \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}, t; \{p\}_\alpha)\; d\Gamma = 0 \qquad \forall \underline{\boldsymbol{u}}^* \in \mathcal{V} \quad \forall t \in]0, T] \quad (2.4)$$

où $\underline{\boldsymbol{u}}^*$ est une fonction test (un champ de déplacements virtuel cinématiquement admissible).

Les modèles élastoplastiques sont largement utilisés pour prédire la fatigue ou la propagation d'une fissure des composants métalliques [Tanguy et al., 2005] [Nicouleau et al., 2002] [Mazière et al., 2009]. De manière courante, la connaissance de l'opérateur $\underset{\sim}{\boldsymbol{\Sigma}}$ est suffisante pour trouver une solution au problème mécanique défini par l'équation 2.4. Les lois de comportement décrivant cet opérateur sont généralement formulées dans le cadre de la thermodynamique des processus irréversibles [Germain et al., 1983]. L'histoire des efforts est pris en compte en utilisant des variables internes. Ces variables sont la somme globale de l'histoire des changements du matériau : elles sont introduites pour tenir compte des phénomènes dissipatifs, dont l'état actuel dépend aussi de l'histoire passée. Cette approche provient des travaux de Biot [Biot, 1965], Ziegler [Ziegler, 1963], Germain [Germain, 1973], ou Halphen et Nguyen [Halphen and Nguyen, 1975], et sa capacité à couvrir un large spectre de modèles en viscoélasticité, viscoplasticité, plasticité a été prouvée. Le lecteur trouvera des exemples de ces lois de comportement en [Lemaitre and Chaboche, 1985]. Cette approche est également largement répandue dans la mécanique continue de la rupture et de l'endommagement [Berdin et al., 2004]. Rappelons que, pour simplifier la présentation des différentes méthodes proposées, nous nous placerons dans le cadre des problèmes mécaniques à petits déplacements, petites déformations. Pour formuler une telle loi de comportement dans le cadre de la thermodynamique des processus irréversibles, il faut définir :

- un potentiel thermodynamique (par exemple, l'énergie libre spécifique $w(\underset{\sim}{\boldsymbol{\varepsilon}}, \underline{\boldsymbol{z}}; \{p\}_\alpha)$) avec ses variables observables $\underset{\sim}{\boldsymbol{\varepsilon}}$ et variables internes $\underline{\boldsymbol{z}}$. Des variables conjuguées $\underline{\boldsymbol{Z}}$ sont associées aux variables internes $\underline{\boldsymbol{z}}$ en utilisant la définition de la dissipation.
- un pseudo potentiel de dissipation φ^* [Germain et al., 1983] (la loi complémentaire) pour décrire l'évolution des variables internes, ou des lois d'évolution reliant $\underline{\boldsymbol{Z}}$ et $\dot{\underline{\boldsymbol{z}}}$.

Les variables conjuguées sont reliées aux variables internes par l'équation d'état suivante :

$$\underline{\boldsymbol{Z}} = -\frac{\partial w}{\partial \underline{\boldsymbol{z}}} \tag{2.5}$$

Des lois complémentaires peuvent être proposées sans introduire le pseudo potentiel de dissipation à condition que l'inégalité de Clausius Duhem soit satisfaite (le taux de création d'entropie doit être non nul) :

$$\dot{\underline{\boldsymbol{z}}} = \underline{\boldsymbol{B}}\,(\underline{\boldsymbol{Z}}\,; \{p\}_{\alpha}). \tag{2.6}$$

Dans le cas de lois de comportements standards, les lois complémentaires sont déduites du pseudo potentiel de dissipation $\varphi^*(\underline{\boldsymbol{Z}}\,; \{p\}_{\alpha})$ telles que :

$$\underline{\boldsymbol{B}}\,(\underline{\boldsymbol{Z}}\,) = \frac{\partial \varphi^*}{\partial \underline{\boldsymbol{Z}}} \tag{2.7}$$

L'état initial du matériau est aussi défini par la condition initiale :

$$\underline{\boldsymbol{z}}_{\,|t=0} = \underline{\boldsymbol{z}}_{\,ini} \tag{2.8}$$

Du fait que dans notre contexte mécanique, la contrainte $\underset{\sim}{\boldsymbol{\sigma}}$ est la variable conjuguée ($\underline{\boldsymbol{Z}}$) à la variable observable $\underset{\sim}{\boldsymbol{\varepsilon}}$ (déformation), les équations (2.5) à (2.8) définissent l'opérateur $\underset{\sim}{\boldsymbol{\Sigma}}$. Ce sont des équations locales : le tenseur de déformation à un point $\underline{\boldsymbol{X}}'$ n'est pas une variable des équations de comportement au point $\underline{\boldsymbol{X}} \neq \underline{\boldsymbol{X}}'$. La formulation de la loi de comportement n'a pas d'incidence sur la définition et la mise en oeuvre des algorithmes étudiés dans cette thèse. De plus, on ne distingue pas les différentes composantes de $\underline{\boldsymbol{z}}$.
On en déduit que les problèmes relatifs à de tels modèles mécaniques sont non linéaires et dépendant du temps.

2.1.1.2 Formulation du modèle Éléments Finis en mécanique non linéaire

En section 1.1.1, nous avons établi le problème de référence à résoudre (1.11) que nous reprenons ici pour des problèmes mécaniques :

On cherche $\underline{\boldsymbol{u}} \in \mathcal{U}_h$ *relatif à un degré de précision* ϵ_h *donné et satisfaisant les équations de comportement du milieu continu et le principe des travaux virtuels, tel que :*

$$\int_{\Omega} \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) \,:\, \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_{\tau}(\underline{\boldsymbol{u}}),\, \tau \leq t; \{p\}_{\alpha})\; d\Omega \;\; - \int_{\partial_f \Omega} \underline{\boldsymbol{u}}^* \,.\, \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}, t; \{p\}_{\alpha})\; d\Gamma = \int_{\Omega} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f \Omega} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b d\Gamma \quad \forall\, \underline{\boldsymbol{u}}^* \in \mathcal{V}_h,\;\; \forall t \in]t_0, t_f] \quad (2.9)$$

Avec la condition suivante :

$$\int_{\Omega} \underline{\boldsymbol{r}}_v . \underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f \Omega} \underline{\boldsymbol{r}}_b . \underline{\boldsymbol{r}}_b d\Gamma \leq \epsilon_h \;\; \forall t \in]0, T] \quad (2.10)$$

Remarque :
ϵ_h est l'erreur induite par l'approximation Éléments Finis. Les équations continues fournissent un résidu, notamment car la contrainte calculée avec la méthode Éléments Finis n'est pas rigoureusement en équilibre (elle est en équilibre au sens faible). En effet, nous utilisons une discrétisation Éléments Finis, donc nous sommes incapables de vérifier rigoureusement toutes les équations du modèle continu. Cette erreur est donc une conséquence d'une discrétisation plus ou moins fine, ce n'est pas un paramètre de la méthode que l'on choisit. Une solution telle que le résidu au sens Éléments Finis soit suffisamment petit sera acceptée.

La formulation des équations réduites diffère de la formulation des équations détaillées que nous venons de voir, par le choix de l'espace de fonctions relatif aux variables d'états, comme nous allons le voir dans la section suivante.

2.1.2 Formulation de l'approximation réduite POD-Galerkin

Le problème réduit à résoudre est :

Trouver $\underline{\boldsymbol{u}} \in \mathcal{U}_{POD}$ *relatif à un degré de précision* $\varepsilon_{cv} = \epsilon_R + \epsilon_h$ *donné et satisfaisant les équations de comportement du milieu continu et le principe des travaux virtuels, tel que :*

$$\int_{\Omega} \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) \,:\, \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_{\tau}(\underline{\boldsymbol{u}}),\, \tau \leq t; \{p\}_{\alpha})\; d\Omega \;\; - \int_{\partial_f \Omega} \underline{\boldsymbol{u}}^* \,.\, \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}, t; \{p\}_{\alpha})\; d\Gamma = \int_{\Omega} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f \Omega} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b d\Gamma \quad \forall\, \underline{\boldsymbol{u}}^* \in \mathcal{V}_{POD},\;\; \forall t \in]t_0, t_f] \quad (2.11)$$

Avec la condition suivante :

$$\int_{\Omega} \underline{\boldsymbol{r}}_v . \underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f \Omega} \underline{\boldsymbol{r}}_b . \underline{\boldsymbol{r}}_b d\Gamma \leq \epsilon_R + \epsilon_h \ \ \forall t \in]0, T] \tag{2.12}$$

Remarque 1 :
En plus du résidu ϵ_h qui apparaît comme conséquence de la discrétisation Éléments Finis, nous tolérons que, le problème étant non-linéaire, cette condition d'équilibre Éléments Finis soit approchée. L'erreur ϵ_R est donc liée à l'approximation en base réduite dont on souhaite maîtriser la qualité. Il s'agit du ratio de convergence de l'algorithme de Newton.
Il est clair que l'existence d'une solution du problème approximatif défini par les équations (2.11-2.12) dépend de la valeur de $\epsilon_{cv} = \epsilon_R + \epsilon_h$ lorsque la solution du problème exact ne peut pas être représentée en utilisant une somme finie, comme proposée lors de l'équation (1.7). Jusqu'à présent, nous n'avons jamais rencontré de problème de convergence. Il se pourrait effectivement que l'on n'arrive pas à atteindre la précision demandée. Mais, puisque l'on utilise un schéma incrémental, et que l'on a une discrétisation en temps, nous obtenons un nombre fini de pas de temps dans notre modèle, et par conséquent, un nombre fini d'états que l'on peut superposer. Par exemple, si l'on a m pas de temps, la solution Éléments Finis est représentable avec m champs Éléments Finis, et donc forcément, la solution discrète en temps est superposable. La condition de qualité sera donc respectée. Toutefois, la solution n'a pas forcément la forme que l'on cherche (1.7). Il faut parfois une somme infinie pour arriver à convergence, et même dans ce cas, la solution n'est pas forcément séparable. Dans ce cas, nous ne pourrons pas atteindre le critère de convergence (2.12). Nous n'aurons donc pas le choix libre de ce seuil ϵ_R, il faudra l'augmenter.
Si toutefois, l'utilisateur souhaite effectuer une simulation en base figée (c'est à dire sans adaptation de la base réduite), il suffira de choisir ϵ_R très grand.

Remarque 2 :
En prenant des fonctions tests $\underline{\boldsymbol{u}}^* \in \mathcal{U}_{h0}$ dans la formulation du principe des puissances virtuelles (2.9) ($\mathcal{U}_{h0}$ étant l'espace des champs cinématiquement admissibles à zéro, représentés par la méthode Éléments Finis), seul le résidu ϵ_R relatif à l'approximation réduite est visible. Il faut pourtant être capable de voir le résidu ϵ_h lié à la méthode Éléments Finis. Pour cela, il faut prendre des fonctions $\underline{\boldsymbol{u}}^* \in \mathcal{U}_0$ (l'ensemble des champs cinématiquement admissibles à zéro au sens des milieux continus).

Comme nous l'avons expliqué précédemment, la qualité du modèle d'ordre réduit peut être vérifiée grâce au résidu $\underline{\boldsymbol{\mathcal{R}}}$ de la condition d'équilibre Éléments

Finis :

$$\underline{\boldsymbol{\mathcal{R}}}_j(\{q\}, t; \{p\}_\alpha) = \int_\Omega \underset{\approx}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{N}}_j) \,:\, \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\approx}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}), \tau \leq t; \{p\}_\alpha) \; d\Omega \quad (2.13)$$
$$- \int_{\partial_f \Omega} \underline{\boldsymbol{N}}_j \cdot \underline{\boldsymbol{f}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha) \; d\Gamma$$

avec

$$\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha) = \sum_{i=1}^{i=n} \underline{\boldsymbol{N}}_i(\underline{\boldsymbol{X}}) \, q_i(t; \{p\}_\alpha) \quad (2.14)$$

2.1.3 Formulation Petrov-Galerkin proposée : la méthode APHR

Pour les deux modèles, le modèle standard Eléments Finis et le modèle réduit POD, nous devons résoudre les équations de comportement locales et non-linéaires pour avoir une estimation du champ de contraintes $\boldsymbol{\Sigma}(\varepsilon_\tau(\underline{\boldsymbol{u}}), \tau \leq t; \{p\}_\alpha)$ en tout point du domaine Ω. Le fait que le déplacement $\underline{\boldsymbol{u}}$ soit pris dans le sous espace de fonctions $\mathcal{V}_{POD}$ n'a aucune incidence sur la complexité des équations locales de la formulation (1.57).

Nous avons vu (section 1.3.2.2) qu'un moyen de réduire le nombre d'équations locales est la création d'un domaine réduit d'intégration Ω_Π. Ce RID Ω_Π est défini comme support de tous les champs tests tronqués :

$$\Omega = \bar{\Omega}_\Pi \cup \Omega_\Pi, \quad (2.15)$$

$$\bar{\Omega}_\Pi = \underline{\boldsymbol{X}} \in \Omega | \forall \{q\}^* \in \mathbb{R}^n, \|\underline{\boldsymbol{u}}^*_\Pi(\underline{\boldsymbol{X}})\| + \|\mathrm{grad}\,(\underline{\boldsymbol{u}}^*_\Pi)(\underline{\boldsymbol{X}})\| = 0. \quad (2.16)$$

où $\|\mathrm{grad}\,(\underline{\boldsymbol{u}}^*_\Pi)\|^2 = \mathrm{grad}\,(\underline{\boldsymbol{u}}^*_\Pi) : \mathrm{grad}\,(\underline{\boldsymbol{u}}^*_\Pi)$.

Pour des problèmes mécaniques, la formulation (1.134) devient :

$$\int_{\Omega_\Pi} \underset{\approx}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) \,:\, \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\approx}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}), \tau \leq t; \{p\}_\alpha) \; d\Omega \; - \int_{\partial_f \Omega_\Pi} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{f}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha) \; d\Gamma =$$
$$\int_{\Omega_\Pi} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f \Omega_\Pi} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b d\Gamma$$
$$\forall \underline{\boldsymbol{u}}^* \in \mathcal{V}_\Pi, \;\; \forall t \in]t_0, t_f], \quad \forall \underline{\boldsymbol{X}} \in \partial_U \Omega \quad \text{avec } \mathbf{u} \in \mathcal{U}_{POD} \quad (2.17)$$

où $\partial_f \Omega_\Pi = \partial_f \Omega \cap \partial \Omega_\Pi$.

Rappelons qu'une fois ces variables réduites calculées, le champ de déplacement est obtenu par l'approximation réduite (1.49). Les variables internes $\underline{\boldsymbol{z}}$ sont, quant à elles, calculées à l'intérieur du RID lors de l'évaluation de $\underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}), \tau \leq t; \{p\}_\alpha)$ mais pas dans le reste du domaine $(\Omega - \Omega_\Pi)$.

2.2 Extrapolation des variables internes

Nous venons d'expliquer que l'on ne calcule les variables internes $\underline{\boldsymbol{z}}$ que sur Ω_Π. Du fait de notre stratégie adaptative et pour pouvoir enchaîner des pas en modèle réduit et des pas en modèle Éléments Finis standards, nous nous devons de pouvoir estimer les variables internes d'un modèle numérique sur tout le domaine Ω. Nous proposons de construire cette estimation en utilisant une extrapolation linéaire des variables internes calculées sur Ω_Π.

Soit $\widehat{\underline{\boldsymbol{z}}}$ les variables internes calculées lors de l'estimation de

$$\underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}})\ ,\ \tau \leq t; \{p\}_\alpha)$$

durant la résolution de (2.17) sur Ω_Π. De la même manière que l'on a défini une approximation réduite des degrés de libertés, nous construisons une base POD à partir des snapshots des variables internes $(\underline{\boldsymbol{z}}_i)_{i=1\ldots m}$ récupérées lors des calculs Éléments Finis (FIG. 2.1).

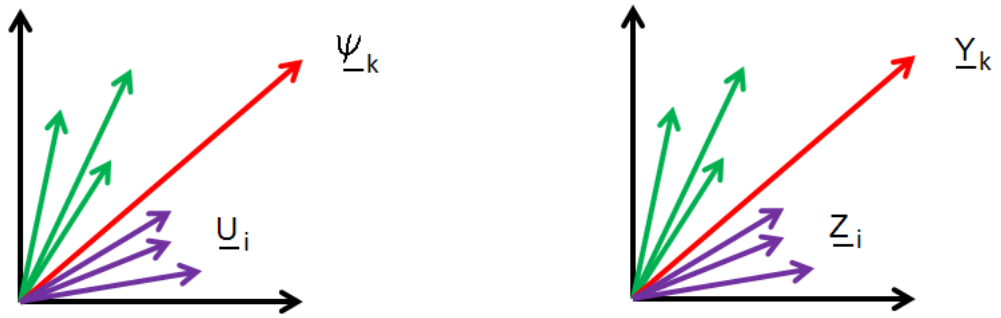


FIG. 2.1 – Définition des bases POD pour les déplacements et les variables internes.

Soit $(\underline{\boldsymbol{Y}}_k)_{k=1\ldots\xi}$ la base POD relative à ces snapshots, elle est définie comme suit :

$$\underline{\boldsymbol{Y}}_k(x) = \sum_{i=1}^{i=m} \underline{\boldsymbol{z}}\,(\underline{\boldsymbol{X}}, t_i; \{p\}_\alpha)\, c_{ik} \quad \forall \underline{\boldsymbol{X}} \in \Omega, \tag{2.18}$$

avec

$$G_{ij} = \int_{\Omega} \underline{\boldsymbol{z}}\left(\underline{\boldsymbol{X}}, t_i; \{p\}_{\alpha}\right) \cdot \underline{\boldsymbol{z}}\left(\underline{\boldsymbol{X}}, t_j; \{p\}_{\alpha}\right) d\Omega \tag{2.19}$$

$$\mu_k = \frac{\{c\}_k^T \cdot [G] \cdot \{c\}_k}{\{c\}_k^T \cdot \{c\}_k} \tag{2.20}$$

$$\sum_{k=1}^{k=\xi} \mu_k > (1 - \varepsilon_{VI}) \sum_{k=1}^{k=m} \mu_k \tag{2.21}$$

Nous retrouvons :

- une matrice $[G]$ de covariance (l'équivalent de $[M]$ pour les degrés de libertés),
- les vecteurs propres $\{c\}_k$ relatifs aux valeurs propres μ_k,
- le seuil de troncature ε_{VI}.

L'approximation multi-niveau des variables internes est la suivante :

$$\underline{\boldsymbol{z}}\left(\underline{\boldsymbol{X}}, t; \{p\}_{\alpha}\right) = \underline{\boldsymbol{z}}_{ROM}(\underline{\boldsymbol{X}}, t; \{p\}_{\alpha}) + \delta\underline{\boldsymbol{z}}\left(\underline{\boldsymbol{X}}, t; \{p\}_{\alpha}\right) \tag{2.22}$$

avec :

$$\underline{\boldsymbol{z}}_{ROM}(\underline{\boldsymbol{X}}, t; \{p\}_{\alpha}) = \sum_{k=1}^{k=\xi} \underline{\boldsymbol{Y}}_k^{(n)}(\underline{\boldsymbol{X}}) b_k(t; \{p\}_{\alpha}) \quad \forall \underline{\boldsymbol{X}} \in \overline{\Omega}_{\Pi}. \tag{2.23}$$

où

$$\underline{\boldsymbol{z}}_{ROM} = \widehat{\underline{\boldsymbol{z}}} \quad \forall \underline{\boldsymbol{X}} \in \Omega_{\Pi}. \tag{2.24}$$

où $\{b\}$ est le vecteur colonne des variables internes réduites.

Les inconnues sont les variables internes réduites $\{b\}\left(t_{i+1}; \{p\}_{\alpha}\right)$ et les variables internes $\underline{\boldsymbol{z}}_{ROM}(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_{\alpha})$ localisées sur $\overline{\Omega}_{\Pi}$. La base POD étant définie sur tout Ω, l'extrapolation se fait comme suit :

$$\{b\}\left(t_{i+1}; \{p\}_{\alpha}\right) = arg\, min_{\{y\}}\, H(\{y\}) \tag{2.25}$$

$$H(\{y\}) = \int_{\Omega_{\Pi}} \left(\widehat{\underline{\boldsymbol{z}}} - \sum_{k=1}^{k=\xi} \underline{\boldsymbol{Y}}_k(\underline{\boldsymbol{X}})\, y_k\right) \cdot \left(\widehat{\underline{\boldsymbol{z}}} - \sum_{k=1}^{k=\xi} \underline{\boldsymbol{Y}}_k(\underline{\boldsymbol{X}})\, y_k\right) d\Omega \tag{2.26}$$

$$\underline{\boldsymbol{z}}_{ROM}(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_{\alpha}) = \sum_{k=1}^{k=\xi} \underline{\boldsymbol{Y}}_k(\underline{\boldsymbol{X}}) b_k(t_{i+1}; \{p\}_{\alpha}) \quad \forall \underline{\boldsymbol{X}} \in \overline{\Omega}_{\Pi} \tag{2.27}$$

Les variables internes réduites $\{b\}$ sont calculées de manière à minimiser l'écart de la projection (2.26) des variables internes $\widehat{\underline{\boldsymbol{z}}}$ calculées sur le RID sur la base $(\underline{\boldsymbol{Y}}_k)_{k=1...\xi}$.

2.3 Fondements de la méthode APHR

Dans la section (2.1), nous avons posé les fondement mathématiques de l'approche. Nous nous attacherons dans cette section à illustrer les choix adoptés pour concrétiser notre stratégie de réduction de modèle. L'approche classique pour l'utilisation de modèles d'ordre réduit consiste en une stratégie à deux étapes (FIG. 2.2) :

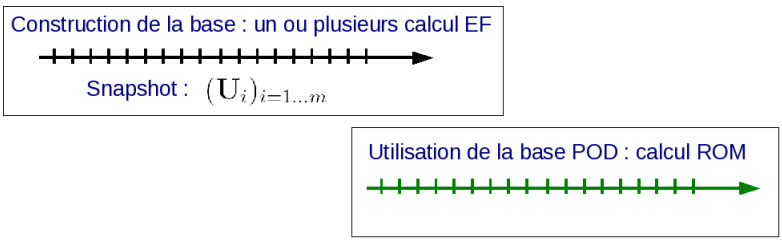


FIG. 2.2 – Stratégie de calcul POD classique.

- On construit une matrice de réduction de base $[A]$, soit par POD ou par d'autres approches. La base réduite est déduite à partir d'un ensemble de snapshots tirés de simulations précédentes, de solutions connues de problèmes similaires.
- On utilise cette connaissance a priori du problème pour accélérer les calculs suivants.

Cette approche a l'énorme inconvénient d'être le traitement a posteriori d'une suite de calculs coûteux pour pouvoir construire la base. De plus, rien ne nous garantit la validité de cette base pour la suite du calcul.

Pour palier à ces inconvénients, on adopte une approche adaptative de la réduction de modèle.

2.3.1 Une stratégie adaptative

Comme nous l'avons déjà précisé, l'approche proposée est une approche multi-niveau. L'approximation multi-niveau est la suivante :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) = \underline{\boldsymbol{u}}\,^{(n)}_{ROM}(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) + \delta\underline{\boldsymbol{u}}\,^{(n)}_{h}(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha), \qquad \underline{\boldsymbol{u}}\,^{(n)}_{ROM} \in \mathcal{V}^{(n)}_{POD}, \quad \delta\underline{\boldsymbol{u}}\,^{(n)}_{h} \in \mathcal{V}_h \tag{2.28}$$

et l'approximation multi-niveau des variables internes est la suivante :

$$\underline{\boldsymbol{z}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) = \underline{\boldsymbol{z}}\,_{ROM}(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) + \delta\underline{\boldsymbol{z}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) \tag{2.29}$$

avec :

$$\underline{\boldsymbol{z}}\,_{ROM}(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) = \sum_{k=1}^{k=\xi} \underline{\boldsymbol{Y}}\,^{(n)}_{k}(\underline{\boldsymbol{X}}\,)b^{(n)}_{k}(t;\{p\}_\alpha). \tag{2.30}$$

où $\{b\}^{(n)}$ est le vecteur colonne des variables internes réduites.

Notation :

Puisqu'il y a une notion d'adaptation, il y a évolution du modèle d'ordre réduit au cours du traitement de la simulation, nous introduirons donc l'exposant (n) qui indiquera la version du modèle d'ordre réduit (soit le nombre d'adaptations qui ont été réaliseés.)

L'indice (n) est donc la version de l'approximation réduite. Une telle approximation multi-niveau permet d'introduire simplement la procédure adaptative. Si $\delta\underline{\boldsymbol{u}}\,^{(n)}_{h}$ et les fonctions de formes $\underline{\boldsymbol{\psi}}\,^{(n)}_{k}(\underline{\boldsymbol{X}}\,)$ du modèle d'ordre réduit ne sont pas linéairement dépendantes, un modèle d'ordre réduit étendu est obtenu tel que $\mathcal{V}^{(n+1)}_{POD} = \text{span}\left\{\underline{\boldsymbol{\psi}}\,^{(n)}_{1}, ..., \underline{\boldsymbol{\psi}}\,^{(n)}_{\widehat{m}}, \delta\underline{\boldsymbol{u}}\,^{(n)}_{h}\right\}$. Ce dernier modèle d'ordre réduit fournit une représentation à variables séparées des déplacements en utilisant les champs définis sur Ω et les fonctions scalaires du temps et des paramètres.

Grâce à un schéma d'intégration numérique à pas unique, on peut estimer différents états du système à différents instants. On suppose l'état mécanique connu au temps t_i. Les inconnues sont les variables détat au temps t_{i+1}. Plusieurs étapes sont nécessaires pour estimer l'état mécanique sur l'incrément de temps $]t_i, t_{i+1}]$:

- Étape 1 : les variables d'état réduites relatives au champ de déplacements sont estimées en supposant $\delta\underline{\boldsymbol{u}}\,^{(n)}_{h} = 0$
- Étape 2 : extrapolation des variables internes de Ω_Π à $\overline{\Omega}_\Pi$
- Étape 3 : l'erreur relative définie à l'équation (1.58) permet d'estimer la précision de l'approximation réduite

- Étape 4 : si l'approximation réduite n'est pas assez précise, une correction Éléments Finis $\delta \underline{\boldsymbol{u}}_h^{(n)}$ est effectuée
- Étape 5 : si la correction Éléments Finis a été effectuée, alors les bases réduites sont adaptées en utilisant les résultats de l'étape de correction (étape 4)
- Étape 6 : reconstruction du RID (que l'on précisera à la section suivante).

Si la prédiction du modèle d'ordre réduit est assez précise, aucun calcul Éléments Finis n'est effectué. En pratique, l'adaptation du modèle d'ordre réduit ne se fait que sur quelques incréments Éléments Finis. De plus, dans le cas d'équations non-linéaires ou de calcul parallèle, le solveur itératif utilise la prédiction fournie par le modèle d'ordre réduit. Il s'agit d'une propriété d'initialisation du modèle d'ordre réduit d'une solution non-linéaire Éléments Finis comme l'on peut le trouver en [Markovinovic and Jansen, 2006]. Ainsi, l'algorithme proposé peut être vu comme un algorithme de Newton-Raphson à deux étapes. Pendant l'étape 1, la prédiction appartient à l'espace des fonctions relatives au modèle d'ordre réduit, tandis qu'à l'étape 3, la correction appartient à l'espace des fonctions Éléments Finis.

À la fin de la quatrième étape, l'état Éléments Finis est connu. Alors l'algorithme proposé en [Ryckelynck et al., 2006] peut être appliqué. L'étape d'adaptation implique l'extension du sous-espace relatif au modèle d'ordre réduit et la sélection des évènements les plus significatifs en utilisant la POD appliquée aux variables d'état réduites. L'extension de la base réduite est effectuée en rajoutant les contributions orthogonales fournies par les corrections Éléments Finis. Les variables d'état réduites relatives aux calculs précédents sont mises à jour pendant l'adaptation du modèle d'ordre réduit. La norme du champ d'effort est définie comme :

$$\|f\|^2 = \int_{\partial_f \Omega_\Pi} \underline{\boldsymbol{f}}\left(\underline{\boldsymbol{X}}, t; \{p\}_\alpha\right).\underline{\boldsymbol{f}}\left(\underline{\boldsymbol{X}}, t; \{p\}_\alpha\right) d\Gamma. \tag{2.31}$$

Les quatre étapes de l'algorithme sont formulées de manière plus détaillée :

Étape 1 :
Les inconnues sont les variables d'état réduites $\{a\}^{(n)}\left(t_{i+1}; \{p\}_\alpha\right)$ et les variables internes $\underline{\boldsymbol{z}}_{ROM}(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_\alpha)$ calculées sur Ω_Π telles que :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha) = \sum_{k=1}^{k=\widehat{m}} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}\,)\,a_k^{(n)}(t;\{p\}_\alpha) + \underline{\boldsymbol{u}}_{ch}(\underline{\boldsymbol{X}}\,,t) \quad (2.32)$$

$$\int_{\Omega_\Pi} \underset{\approx}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) \,:\, \underset{\approx}{\boldsymbol{\Sigma}}(\underset{\approx}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}\,),\,\tau\le t;\{p\}_\alpha)\;d\Omega \;-\int_{\partial_f\Omega_\Pi} \underline{\boldsymbol{u}}^*\,.\,\underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha)\;d\Gamma =$$
$$\int_{\Omega_\Pi} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f\Omega_\Pi} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_b d\Gamma \quad \forall\,\underline{\boldsymbol{u}}^* \in \mathcal{V}_\Pi^{(n)},\;\; \forall t\in\,]t_0,t_f] \quad (2.33)$$

Étape 2 :
Extrapolation des variables internes de Ω_Π à $\overline{\Omega}_\Pi$. Le modèle réduit étant adapté, l'extrapolation se fait comme suit :

$$\{b\}^{(n)}\,(t_{i+1};\{p\}_\alpha) = arg\,min_{\{y\}}\,H(\{y\}) \quad (2.34)$$

$$H(\{y\}) = \int_{\Omega_\Pi} \left(\widehat{\underline{\boldsymbol{z}}} - \sum_{k=1}^{k=\xi} \underline{\boldsymbol{Y}}_k^{(n)}(\underline{\boldsymbol{X}}\,)\,y_k\right)^2 d\Omega \quad (2.35)$$

$$\underline{\boldsymbol{z}}_{ROM}(\underline{\boldsymbol{X}}\,,t_{i+1};\{p\}_\alpha) = \sum_{k=1}^{k=\xi} \underline{\boldsymbol{Y}}_k^{(n)}(\underline{\boldsymbol{X}}\,)b_k^{(n)}(t_{i+1};\{p\}_\alpha) \quad \forall \underline{\boldsymbol{X}}\,\in\overline{\Omega}_\Pi \quad (2.36)$$

Étape 3 :
Évaluation de l'indicateur d'erreur η_{POD} relatif au résidu tronqué $\left\{\tilde{\mathcal{R}}\right\}$:

$$\left\{\tilde{\mathcal{R}}\right\}_j \;=\; \int_{\Omega_\Pi} \underset{\approx}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) \,:\, \underset{\approx}{\boldsymbol{\Sigma}}(\underset{\approx}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}\,),\,\tau\le t;\{p\}_\alpha)\;d\Omega \;-\int_{\partial_f\Omega_\Pi} \underline{\boldsymbol{u}}^*\,.\,\underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha)\;d\Gamma \quad (2.37)$$

avec

$$\begin{aligned} \underline{\boldsymbol{u}}^* &= \underline{\boldsymbol{N}}_j(\underline{\boldsymbol{X}}\,)(\max_i \Pi_{ij}) \\ \eta_{POD} &= \|\left\{\tilde{\mathcal{R}}\right\}\| \end{aligned}$$

Étape 4 :
- Si $\eta_{POD} < \epsilon_R\|f\|$, alors $\quad \delta\underline{\boldsymbol{u}}_h(\underline{\boldsymbol{X}}\,,t_{i+1};\{p\}_\alpha) = 0$,
- Sinon, la correction des déplacements est telle que :

$$\int_\Omega \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^* + \delta\underline{\boldsymbol{u}}_h^{(n)}) \,:\, \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}_{POD}^{(n)} + \delta\underline{\boldsymbol{u}}_h^{(n)}) \,,\, \tau \leq t; \{p\}_\alpha) \;\; d\Omega$$
$$- \int_{\partial_f\Omega} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha) \;\; d\Gamma = \int_\Omega \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f\Omega} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b d\Gamma \tag{2.38}$$
$$\forall\, \underline{\boldsymbol{u}}^* \in \mathcal{V}_h, \;\; \forall t \in]t_0, t_f]$$

Étape 5 :
Si le critère de validité de la base n'est pas satisfait, (*i.e* $\eta_{POD} \geq \epsilon_R \|f\|$), alors le sous-espace $\mathcal{V}_{POD}^{(n)}$ est adapté en utilisant $\delta\underline{\boldsymbol{u}}_h^{(n)}$.
Premièrement, la base est étendue, puis, une POD est effectuée sur les variables d'état du modèle d'ordre réduit. La base intermédiaire est notée $\left(\underline{\boldsymbol{\psi}}_k^{(n+1/2)}\right)_{k=1\ldots s+1}$.
Nous ne considèrerons que le résidu relatif à la projection orthogonale de $\delta\underline{\boldsymbol{u}}_h^{(n)}$ sur $\mathcal{V}_{POD}^{(n)}$. Ce résidu est noté $\delta_\perp\underline{\boldsymbol{u}}_h^{(n)}$ et est tel que :

$$\delta_\perp\underline{\boldsymbol{u}}_h^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) = \delta\underline{\boldsymbol{u}}_h^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) - \widehat{\delta\underline{\boldsymbol{u}}_h}^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) \tag{2.39}$$

$$\widehat{\delta\underline{\boldsymbol{u}}_h}^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}\,)\, \delta a_k^{(n)}(t_{i+1}; \{p\}_\alpha) \tag{2.40}$$

$$\{\delta a\}^{(n)}(t_{i+1}; \{p\}_\alpha) = \arg\min_{\{y\}} \int_\Omega \left\| \delta\underline{\boldsymbol{u}}_h(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) - \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}\,)\, y_k \right\|^2 d\Omega \tag{2.41}$$

Alors, la nouvelle base étendue $\left(\underline{\boldsymbol{\psi}}_k^{(n+1/2)}\right)_{k=1\ldots s+1}$ est construite en conservant les prévisions précédentes telles que :

$$\underline{\boldsymbol{\psi}}_k^{(n+1/2)} = \underline{\boldsymbol{\psi}}_k^{(n)} \quad k \leq s \tag{2.42}$$
$$\underline{\boldsymbol{\psi}}_{s+1}^{(n+1/2)} = \frac{1}{\|\delta_\perp\underline{\boldsymbol{u}}_h\|}\, \delta_\perp\underline{\boldsymbol{u}}_h \tag{2.43}$$
$$a_k^{(n+1/2)}(\tau; \{p\}_\alpha) = a_k^{(n)}(\tau; \{p\}_\alpha) \quad \tau \leq t_{i+1} \quad k \leq s \tag{2.44}$$
$$a_{s+1}^{(n+1/2)}(\tau; \{p\}_\alpha) = 0 \quad \tau \leq t_i \tag{2.45}$$
$$a_{s+1}^{(n+1/2)}(t_{i+1}; \{p\}_\alpha) = \|\delta_\perp\underline{\boldsymbol{u}}_h\| \tag{2.46}$$

On effectue une décomposition POD sur les variables réduites afin d'éviter une évolution constante de la taille du modèle d'ordre réduit. Cette POD

consiste à trouver les vecteurs $(\{V\}_l)_{l=1...s+1}$ qui maximisent la projection sur les variables d'état réduites suivante :

$$\lambda_l^{(n+1)} = \frac{\int_0^{t_{j+1}} \left(\{a\}^{(n+1/2)T}(t;\{p\}_\alpha) \,.\, \{V\}_l \right)^2 dt}{\|\{V\}_l\|^2} \tag{2.47}$$

sous les conditions :

$\|\{V\}_l\| = 1$ et $\lambda_l \geq \lambda_{l+1}$.

Les vecteurs $(\{V\}_l)_{l=1...s+1}$ sont les vecteurs propres de la matrice de covariance $[C]^{(n+1/2)}$ telle que :

$$[C]^{(n+1/2)} = \int_0^{t_{j+1}} \{a\}^{(n+1/2)}(t;\{p\}_\alpha) \,.\, \{a\}^{(n+1/2)T}(t;\{p\}_\alpha)\, dt \tag{2.48}$$

La matrice de covariance prend en compte les résultats précédents afin de conserver la capacité du modèle d'ordre réduit à modéliser les états mécaniques. Alors, les évènements les plus significatifs sont sélectionnés en utilisant les critères suivants :

$$[V] = [\{V\}_1, ..., \{V\}_{\widetilde{s}}] \quad \text{avec } \lambda_{\widetilde{s}} > \varepsilon_{POD}\, \lambda_1 \quad \text{et } \lambda_{\widetilde{s}+1} \leq \varepsilon_{POD}\, \lambda_1 \tag{2.49}$$

où $\widetilde{s}$ est la nouvelle taille du modèle d'ordre réduit. Finalement, le nouveau modèle d'ordre réduit est une réduction POD du modèle d'ordre réduit précédent :

$$\underline{\boldsymbol{\psi}}_l^{(n+1)} = \sum_{k=1}^{k=s+1} \underline{\boldsymbol{\psi}}_k^{(n+1/2)}\, V_{kl} \quad \forall l = 1...\widetilde{s} \tag{2.50}$$

$$a_l^{(n+1)}(t;\{p\}_\alpha) = \sum_{k=1}^{k=s+1} a_k^{(n+1/2)}(t;\{p\}_\alpha)\, V_{kl} \quad \forall l = 1...\widetilde{s} \tag{2.51}$$

À la fin de chaque simulation, on effectue une dernière POD sur les variables d'état réduites. Les paramètres de la méthode sont les coefficients ϵ_R et ε_{POD}. Le modèle d'ordre réduit initial $((\underline{\boldsymbol{\psi}}_k^{(0)}, a_k^{(0)})_{k=1...s})$ peut être vide ($s = 0$). Il n'y a pas de connaissance a priori requise. On commence notre calcul incrémental par un ou plusieurs pas de temps en calculs Éléments Finis standards qui fournissent un premier ensemble de snapshots (étape 3). Étant donné que l'on utilise l'algorithme adaptatif [Ryckelynck et al., 2006], la propriété suivante est satisfaite :

Propriété

Dans le cas d'une intégration complète ($\Omega_\Pi = \Omega$), la décomposition obtenue à la fin du calcul incrémental est une POD avec une précision relative à un seuil ϵ_R. Celui-ci peut être choisi de manière à ce que l'on veuille une qualité précise, ou alors un calcul rapide mais de qualité moindre.

On peut trouver la preuve de cette propriété en [Ryckelynck et al., 2006]. Quand un calcul est effectué sans correction (étape 3), l'algorithme est un algorithme classique POD de prévision de l'état mécanique. De plus, on a les trois propriétés suivantes.

Propriétés

1. Si la précision du modèle d'ordre réduit n'est pas assez bonne, ou si ϵ_R est trop élevé, l'algorithme multi-niveau APHR fournit une solution classique relative au modèle d'ordre réduit sans adaptation.
2. Si ϵ_R est très petit, la prédiction du modèle d'ordre réduit est corrigée à chaque incrément de temps, donc l'algorithme multi-niveau APHR fournit une solution Éléments Finis.
3. Si la taille du modèle d'ordre réduit est petite (si ε_{POD} est suffisamment grand), la prévision du modèle d'ordre réduit n'a pas d'effet sur la correction Éléments Finis.

Cette approche a l'avantage de réduire les calculs coûteux au strict minimum, le calcul d'un snapshot par Eléments Finis n'ayant lieu que si celui-ci n'est pas représentable dans la base existante.

Remarque :

Dans le cas d'une intégration complète ($\Omega_\Pi = \Omega$), la méthode est une méthode de POD adaptative que l'on a appelée APR en section 1.3. En effet, le calcul en modèle d'ordre réduit est effectué sur le domaine entier Ω, donc l'originalité "Hyper Réduction" n'est pas exploitée. Les deux sections suivantes décrivent le processus d'Hyper Réduction.

2.4 Construction du RID

Comme évoqué dans la section 1.3.2.2, l'une des originalités majeures de cette approche est l'introduction d'un domaine réduit d'intégration (RID). Dans cette section, on s'attachera à définir la stratégie pour construire un RID selon [Ryckelynck and Missoum-Benziane, 2010], efficace en terme de gains de performance en limitant au maximum l'amplification d'erreur.

Le RID est formé par les éléments connectés aux degrés de libertés relatifs aux équations d'équilibre sélectionnées pour la formulation (2.17). Aussi, la première étape de la construction d'un RID est le choix de ces équations pour que le problème soit bien posé.

On choisit ces équations via un processus itératif.

Soit Ω le maillage tel que $\Omega = \bigcup_{e_i=1}^{N_e} \Omega_{e_i}$.

- Afin d'éviter la solution $\underline{\boldsymbol{u}} = 0$, on crée un premier set de s éléments relatif aux s plus grandes valeurs du résidu initial $|\{\mathcal{R}\}(\{q\}, 0; \{p\}_\alpha)|$. Ceci nous assure la "visibilité" du chargement extérieur par notre formulation et constitue notre RID initial $\mathcal{L}_g^{(0)}$.
- On peut spécifier des régions "d'intérêt particulier" que l'on ajoute à ce premier set $\mathcal{L}_g^{(0)}$.
- Afin d'inclure le maximum de l'énergie du problème dans notre formulation, nous allons inclure les régions où se passent les transformations les plus significatives. Les vecteurs de la base $(\underline{\boldsymbol{Y}}_k)_{k=1\ldots\xi}$ représentant ces transformations pour les variables internes, on leur associe une pseudo énergie libre comme indicateur de l'importance de ces transformations, et nous choisirons donc les ξ éléments où cette énergie atteint son maximum, *i.e* :

$$e_j = \arg\max_{e_i} \int_{\Omega_{e_i}} \underline{\boldsymbol{Y}}_k \cdot \underline{\boldsymbol{Y}}_k d\Omega \tag{2.52}$$

  Nous obtenons ainsi la version $\mathcal{L}_g^{(1)}$ du RID.
- Nous pouvons aussi, selon la nature du problème à traiter, choisir pour chaque vecteur $\underline{\boldsymbol{\psi}}_k$ l'élément où se localise le maximum de la déformation, soit :

$$e_j = \arg\max_{e_i} \int_{\Omega_{e_i}} \underline{\underline{\boldsymbol{\varepsilon}}}(\underline{\boldsymbol{\psi}}_k) \cdot \underline{\underline{\boldsymbol{\varepsilon}}}(\underline{\boldsymbol{\psi}}_k) d\Omega \tag{2.53}$$

  Nous choisirons alors les d éléments supplémentaires et ceci constitue la version $\mathcal{L}_g^{(2)}$ du RID.

Ces premières étapes se traduisent par une localisation des régions d'intérêt et ne définissent pas l'étendue du RID. Pour cela, l'extension du RID peut se faire par l'ajout des éléments connectés à ceux déjà sélectionnés via un paramètre n_c. Ceci constitue la version finale $\mathcal{L}_g^{(3)}$ du RID.

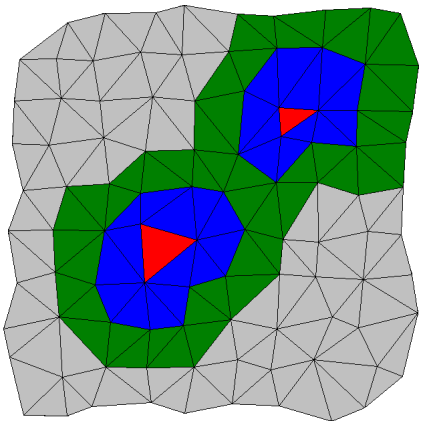


FIG. 2.3 – Extension du RID par le paramètre n_c.

La figure (FIG. 2.3) donne un exemple de l'utilisation du paramètre n_c. Les éléments en rouge représentent le RID originel donné par les critères de maximum d'énergie précités. Les éléments en bleu sont les voisins directs des rouges (se partageant au moins un noeud). L'ensemble (bleu, rouge) forme le RID que l'on obtient avec un paramètre $n_c = 1$. De la même manière, les éléments en vert sont la seconde couche de voisins et leur ajout donne le RID que l'on obtient avec un paramètre $n_c = 2$.

2.5 Application de la méthode APHR

Le problème de l'éprouvette est un grand classique en mécanique, il nous permet de voir le comportement de la méthode dans un cas de forte concentration des contraintes et des déformations. En s'éloignant ainsi des problèmes à solution régulière, on met à l'épreuve les capacités de l'approche adaptative. En effet, une approche POD classique qui ne contient pas toutes les informations ne marcherait pas dans ce cas. Les dimensions en mm de l'éprouvette sont données sur la figure suivante (FIG. 2.4).

Grâce aux conditions de symétrie, nous ne maillerons qu'un huitième de cette éprouvette : pour cela, on impose des conditions de symétrie sur les

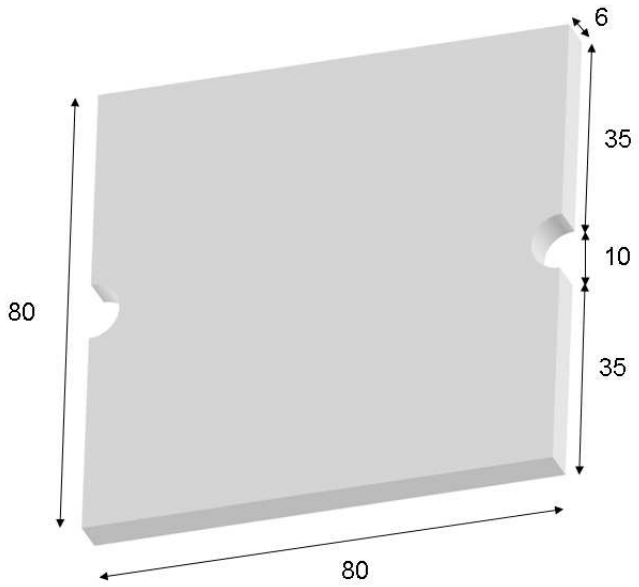


FIG. 2.4 – Dimensions en mm de l'éprouvette.

bords X_{max}, Y_{min} et Z_{min} (les déplacements selon les normales (resp. U_1, U_2 et U_3 sont nuls). Une pression $P(t)$ est imposée sur la surface Y_{max}. La courbe de cette pression en fonction du temps sera donnée pour chaque cas étudié. Le problème est représenté sur la figure (FIG. 2.5), ainsi que sa discrétisation avec les conditions de symétrie.

Le maillage est composé de 200 éléments quadratiques à 20 noeuds à intégration réduite (type c3d20r), et de 1553 noeuds.

Le modèle de comportement est élastoplastique avec écrouissage isotrope linéaire dont la fonction seuil s'écrit à l'aide de la contrainte de von Mises σ_{Mises} de la manière suivante.

$$f(R, \sigma_{Mises}) = \sigma_{Mises} - R, \text{ où } R = Hp + R_0 \tag{2.54}$$

où R_0 est la limite d'élasticité, p la plasticité cumulée, et H le module plastique. Le vecteur paramètres contient les paramètres du matériau : le module d'Young E, le coefficient de poisson ν, le module plastique H et la limite d'élasticité R_0. Ceux-ci sont définis dans le tableau TAB. 2.1.

Sur cette éprouvette, nous imposons une pression $P(t) = 128.8t$.
Les paramètres de la méthode sont les coefficients $\epsilon_R = 0.05$ et $\varepsilon_{POD} = 10^{-8}$.

Le domaine d'intégration réduit est constitué de tous les éléments en rouge que l'on peut voir sur la figure FIG. 2.6

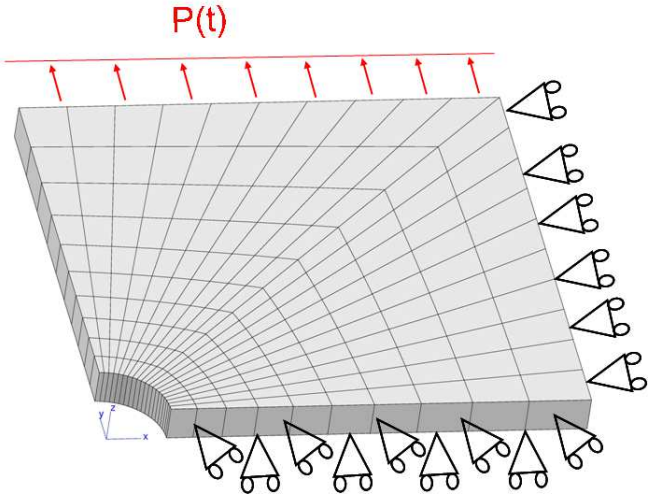


FIG. 2.5 – Maillage d'un huitième de l'éprouvette et conditions aux limites.

$***behavior \;\; gen\_evp$
$** elasticity \;\; isotropic$
$young \;\; 1.987515139180844e+05$
$poisson \;\; 0.3$
$** potential \;\; gen\_evp \;\; ep$
$*flow \;\; plasticity$
$*criterion \;\; mises$
$*isotropic \;\; linear$
$R0 \;\; 1.509522960403531e+02$
$H \;\; 1.923844712160604e+04$
$***return$

TAB. 2.1 – Définition dans le code de calcul ZéBuLoN des paramètres matériau du modèle.

Nous pouvons trouver en figure (FIG. 2.7) un exemple des sept modes empiriques, et en figure (FIG. 2.8), les cinq modes correspondant aux variables internes obtenus après une première simulation en modèle d'ordre réduit sur ce modèle.

Les courbes de l'effort en fonction du déplacement obtenues pour la simulation de référence Éléments Finis et la simulation APHR sont représentées en figure FIG. 2.9.

Remarque

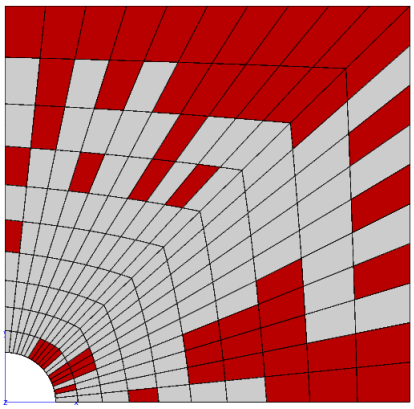


FIG. 2.6 – Domaine d'Intégration Réduit.

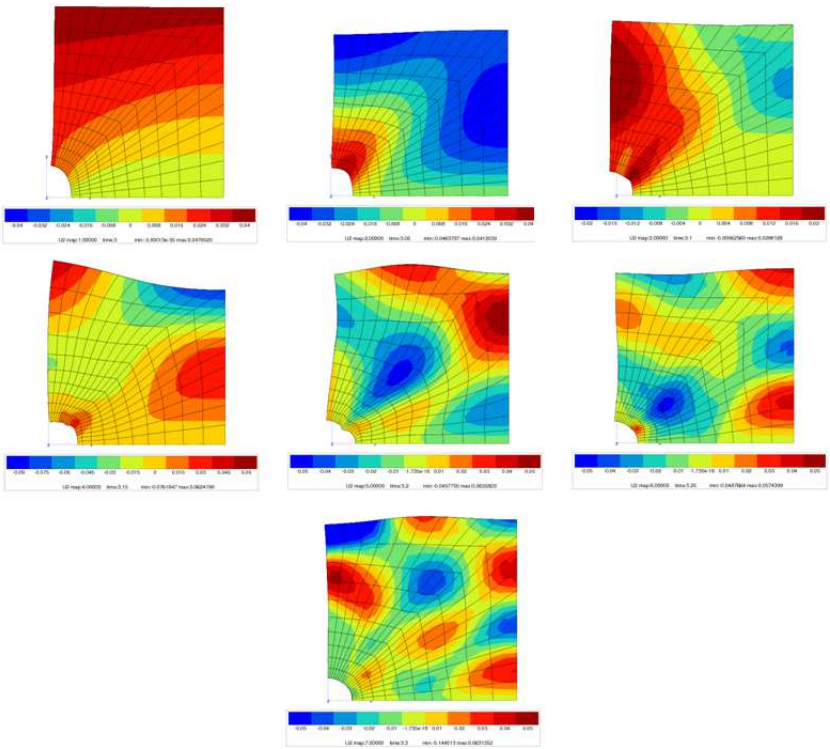


FIG. 2.7 – 7 modes empiriques obtenus après une simulation APHR.

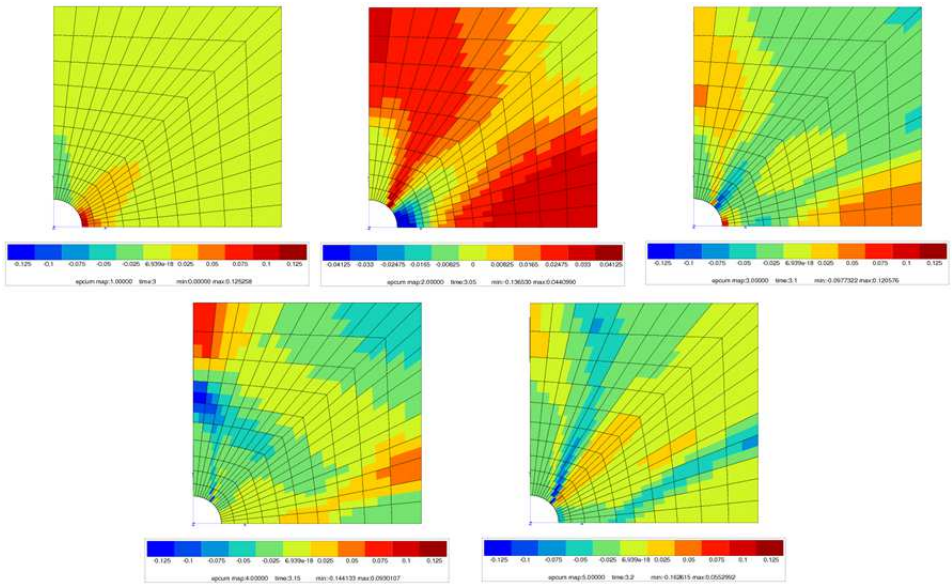


FIG. 2.8 – 5 modes correspondant aux variables internes obtenus après une simulation APHR.

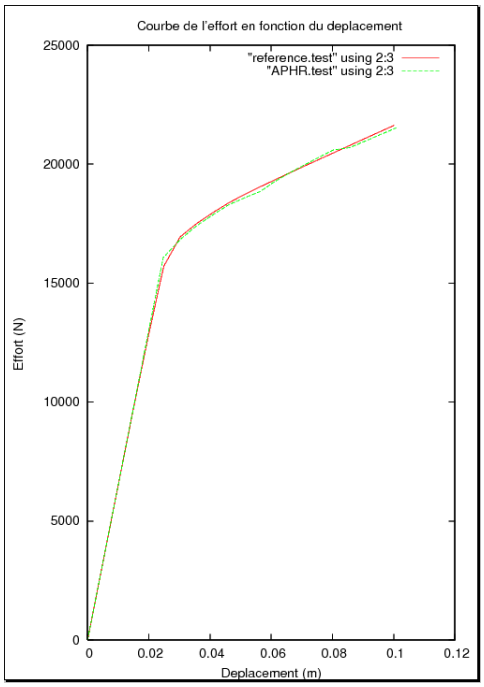


FIG. 2.9 – Courbes effort-déplacement.

Ce problème servira de fil conducteur tout au long du mémoire afin d'illustrer les méthodes que nous proposerons. Dans les chapitres suivants, les valeurs de R_0 et H changeant en fonction du type de simulation, nous préciserons pour chaque cas les valeurs utilisées. De même, les courbes de traction $\sigma = f(\varepsilon)$ seront données pour chaque cas.

Chapitre 3

Modèle d'ordre réduit évolutif pour une suite de simulations

Soit une suite de m simulations $(S_1, S_2, ..., S_m)$ permettant d'obtenir une suite de solutions $(\underline{\boldsymbol{u}}_1, \underline{\boldsymbol{u}}_2, ..., \underline{\boldsymbol{u}}_m)$ représentant l'état du système à différents instants, pour des valeurs du vecteur des paramètres $\{p\}_1$,...,$\{p\}_m$.

Effectuer une suite de simulations consiste à tenir compte des résultats antérieurs (de $\underline{\boldsymbol{u}}_1$ à $\underline{\boldsymbol{u}}_{m-1}$) pour calculer une approximation de $\underline{\boldsymbol{u}}_m$ à l'aide d'une base réduite composée des vecteurs $(\underline{\boldsymbol{\psi}}^{(n)}{}_k)_{k=1...s}$.

Les résultats antérieurs sont exploités pour réduire le coût de la simulation restant à faire (Figure 3.1). Nous proposons de construire, au cours de la suite de simulations, un modèle d'ordre réduit évolutif, sans connaître de base réduite au commencement de la suite de simulations.

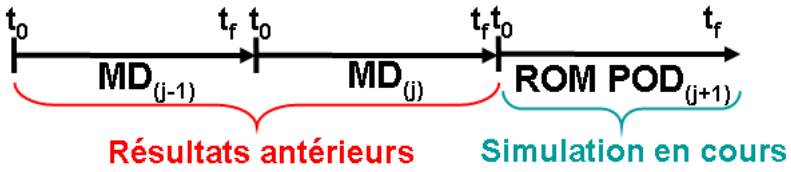


FIG. 3.1 – Suite de simulations produite avec le Modèle Détaillé.

Remarques

- Dans un premier temps, nous nous limitons à l'étude formelle d'une suite de simulations sans que cette suite soit nécessairement relative à un processus d'optimisation.
- Dans un second temps, nous traiterons une suite de simulations dans le cadre de la résolution de problèmes inverses.

3.1 Définition d'un temps global

Suite à un premier calcul Éléments Finis (Modèle Détaillé, noté MD), nous obtenons une première réponse $\underline{\boldsymbol{u}}_1(\underline{\boldsymbol{X}}, t, \{p\}_1)$. Nous pouvons effectuer un second calcul Modèle Détaillé qui donne une seconde réponse $\underline{\boldsymbol{u}}_2(\underline{\boldsymbol{X}}, t, \{p\}_2)$. Une suite de calculs est alors une suite de Modèles Détaillés. Une réponse du système est ainsi simulée. La Figure FIG. 3.2 représente une suite de calculs produite en cours de processus d'optimisation avec une modification de paramètres.

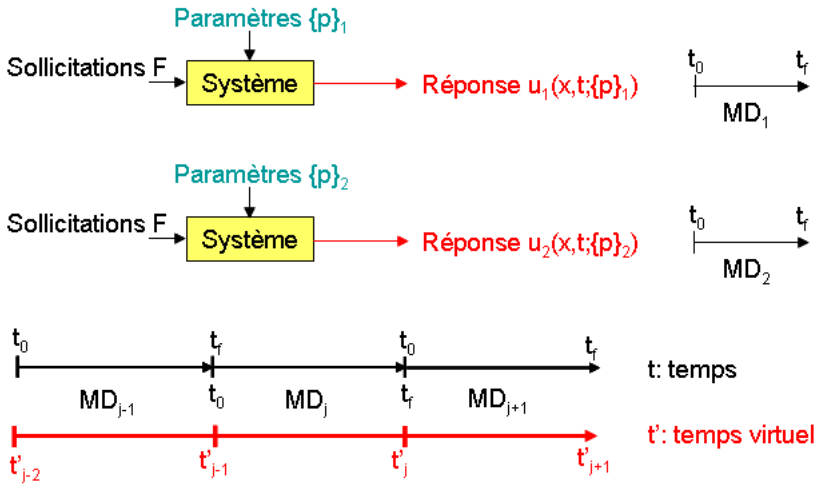


FIG. 3.2 – Suite de calculs en cours de processus d'optimisation.

L'objectif est de traiter l'ensemble des résultats de simulations $\underline{\boldsymbol{u}}_1$ à $\underline{\boldsymbol{u}}_{m-1}$ comme étant un résultat séquentiel global. Il nous semble donc judicieux d'introduire le concept d'un temps global tel que :

$$\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t') = \underline{\boldsymbol{u}}_j(\underline{\boldsymbol{X}}, t' - t'_{j-1}, \{p\}_j), \quad \forall t' \in]t'_{j-1}, t'_j] \tag{3.1}$$

avec $t'_j = t'_{j-1} + T$ où $T = t_f - t_0$ est la durée d'une simulation.

Remarques

- Une suite de calculs conduit donc à un axe de temps virtuel : l'interprétation de cette suite de simulations.
- La continuité par rapport à la variable t' n'est pas nécessaire pour mettre en œuvre une méthode de réduction de modèles.

Notation

Nous allons coupler les notations $X^{(n)}$ (qui décrit une version de l'état X due à l'adaptation), et X_m (qui décrit un état X lors de la $m^{\text{ième}}$ simulation), par la notation $X^{(n,m)}$.

Remarques

Nous avons maintenant le choix entre deux notations pour écrire les réponses du système :

$$\underline{\boldsymbol{u}}_m(\underline{\boldsymbol{X}}, t; \{p\}_m) = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}) \, a_k^{(n,m)}(t; \{p\}_m) \tag{3.2}$$

Ou, plus simplement, grâce à l'introduction d'un temps global :

$$\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t') = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}) \, a_k^{(n)}(t') \tag{3.3}$$

Ainsi reformulé, nous pouvons directement appliquer la méthode APHR à une suite de simulations.

3.2 Méthode APHR appliquée à une suite de simulations

Du fait de la procédure adaptative contenue dans la méthode APHR, nous obtenons une base qui évolue au cours de la suite de simulations. La version de la base (n) est donc liée à m, le nombre de simulations réalisées. Les étapes de la méthode APHR décrites en section 2.3.1 sont reformulées ci-après pour une suite de α simulations réalisées, en tenant compte des $\alpha - 1$ simulations précédentes. Seules les équations (2.47) et (2.48) ont été modifiées, et les équations (3.17) et (3.18) ont été rajoutées.

Étape 1 :

Estimation des variables détat réduites.

$$\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha) = \sum_{k=1}^{k=\widehat{m}} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}) \, a_k^{(n)}(t; \{p\}_\alpha) + \underline{\boldsymbol{u}}_{ch}(\underline{\boldsymbol{X}}, t) \tag{3.4}$$

$$\int_{\Omega_\Pi} \underline{\underline{\boldsymbol{\varepsilon}}}(\underline{\boldsymbol{u}}^*) \, : \, \underline{\underline{\boldsymbol{\Sigma}}}(\underline{\underline{\boldsymbol{\varepsilon}}}_\tau(\underline{\boldsymbol{u}}), \, \tau \leq t; \{p\}_\alpha) \; d\Omega \; - \int_{\partial_f \Omega_\Pi} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{f}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha) \; d\Gamma = \int_{\Omega_\Pi} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f \Omega_\Pi} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b d\Gamma$$
$$\forall \, \underline{\boldsymbol{u}}^* \in \mathcal{V}_\Pi^{(n)}, \;\; \forall t \in]t_0, t_f] \tag{3.5}$$

Étape 2 :
Extrapolation des variables internes de Ω_Π à $\overline{\Omega}_\Pi$.

$$\{b\}^{(n)}\,(t_{i+1};\{p\}_\alpha) = arg\,min_{\{y\}}\,H(\{y\}) \tag{3.6}$$

$$H(\{y\}) = \int_{\Omega_\Pi} < \widehat{\mathbf{z}} - \sum_{k=1}^{k=\xi} \mathbf{Y}_k^{(n)}(\underline{\boldsymbol{X}})\,y_k\;,\;\widehat{\mathbf{z}} - \sum_{k=1}^{k=\xi} \mathbf{Y}_k^{(n)}(\underline{\boldsymbol{X}})\,y_k > \;d\Omega \tag{3.7}$$

$$\mathbf{z}_{ROM}(\underline{\boldsymbol{X}}\,,t_{i+1};\{p\}_\alpha) = \sum_{k=1}^{k=\xi} \mathbf{Y}_k^{(n)}(\underline{\boldsymbol{X}})b_k^{(n)}(t_{i+1};\{p\}_\alpha) \quad \forall \underline{\boldsymbol{X}} \in \overline{\Omega}_\Pi \quad . \tag{3.8}$$

Étape 3 :
Évaluation de l'indicateur d'erreur η_{POD} relatif au résidu tronqué $\underline{\tilde{\boldsymbol{\mathcal{R}}}}$:

$$\underline{\tilde{\boldsymbol{\mathcal{R}}}}_j \;=\; \int_{\Omega_\Pi} \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) \;:\; \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}})\,,\,\tau \le t;\{p\}_\alpha)\;\;d\Omega \;\; - \int_{\partial_f\Omega_\Pi} \underline{\boldsymbol{u}}^* \,.\, \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha)\;\;d\Gamma \tag{3.9}$$

avec

$$\begin{aligned} \underline{\boldsymbol{u}}^* &= \underline{\boldsymbol{N}}\,(\underline{\boldsymbol{X}})(\max_i \Pi_{ij}) \\ \eta_{POD} &= \|\underline{\tilde{\boldsymbol{\mathcal{R}}}}\,\| \end{aligned}$$

Étape 4 :
- Si $\eta_{POD} < \epsilon_R\|f\|$, alors $\quad \delta\underline{\boldsymbol{u}}_h(\underline{\boldsymbol{X}}\,,t_{i+1};\{p\}_\alpha) = 0$,
- Sinon, la correction des déplacements est telle que :

$$\begin{aligned} &\int_\Omega \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*) + \delta\underline{\boldsymbol{u}}_h^{(n)}) \;:\; \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}_{POD}^{(n)} + \delta\underline{\boldsymbol{u}}_h^{(n)})\,,\,\tau \le t;\{p\}_\alpha)\;\;d\Omega \\ &- \int_{\partial_f\Omega} \underline{\boldsymbol{u}}^* \,.\, \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_\alpha)\;\;d\Gamma = \int_\Omega \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f\Omega} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_b d\Gamma \\ &\qquad\qquad \forall\,\underline{\boldsymbol{u}}^* \in \mathcal{V}_h,\;\; \forall t \in\,]t_0,t_f] \end{aligned} \tag{3.10}$$

Étape 5 :

$$\delta_{\perp}\underline{\boldsymbol{u}}_h^{(n)}(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_\alpha) = \delta\underline{\boldsymbol{u}}_h^{(n)}(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_\alpha) - \widehat{\delta\underline{\boldsymbol{u}}_h}^{(n)}(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_\alpha) \tag{3.11}$$

$$\widehat{\delta\underline{\boldsymbol{u}}_h}^{(n)}(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_\alpha) = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}})\, \delta a_k^{(n)}(t_{i+1}; \{p\}_\alpha) \tag{3.12}$$

$$\{\delta a\}^{(n)}(t_{i+1}; \{p\}_\alpha) = \arg\min_{\{y\}} \int_\Omega \left\| \delta\underline{\boldsymbol{u}}_h(\underline{\boldsymbol{X}}, t_{i+1}; \{p\}_\alpha) - \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}})\, y_k \right\|^2 d\Omega \tag{3.13}$$

Alors, la nouvelle base étendue $\left(\underline{\boldsymbol{\psi}}_k^{(n+1/2)}\right)_{k=1\ldots s+1}$ est construite en conservant les prévisions précédentes telles que :

$$\underline{\boldsymbol{\psi}}_k^{(n+1/2)} = \underline{\boldsymbol{\psi}}_k^{(n)} \quad k \leq s \tag{3.14}$$

$$\underline{\boldsymbol{\psi}}_{s+1}^{(n+1/2)} = \frac{1}{\|\delta_{\perp}\underline{\boldsymbol{u}}_h\|}\, \delta_{\perp}\underline{\boldsymbol{u}}_h \tag{3.15}$$

$$a_k^{(n+1/2)}(\tau; \{p\}_\alpha) = a_k^{(n)}(\tau; \{p\}_\alpha) \quad \tau \leq t_{i+1} \quad k \leq s \tag{3.16}$$

$$a_k^{(n+1/2)}(\tau; \{p\}_\beta) = a_k^{(n)}(\tau; \{p\}_\beta) \quad \tau \leq T \quad k \leq s \quad \beta < \alpha \tag{3.17}$$

$$a_{s+1}^{(n+1/2)}(\tau; \{p\}_\beta) = 0 \quad \tau \leq T \quad \beta < \alpha \tag{3.18}$$

$$a_{s+1}^{(n+1/2)}(\tau; \{p\}_\alpha) = 0 \quad \tau \leq t_i \tag{3.19}$$

$$a_{s+1}^{(n+1/2)}(t_{i+1}; \{p\}_\alpha) = \|\delta_{\perp}\underline{\boldsymbol{u}}_h\| \tag{3.20}$$

On effectue une décomposition POD sur les variables réduites afin d'éviter une évolution constante de la taille du modèle d'ordre réduit. Cette POD consiste à trouver les vecteurs $(\{V\}_l)_{l=1\ldots s+1}$ qui maximisent la projection sur les variables d'état réduites suivante :

$$\lambda_l^{(n+1)} = \frac{\sum_{\beta=1}^{\beta=\alpha-1} \int_0^T \left(\{a\}^{(n+1/2)T}(t; \{p\}_\beta)\,.\,\{V\}_l\right)^2 dt + \int_0^{t_{j+1}} \left(\{a\}^{(n+1/2)T}(t; \{p\}_\alpha)\,.\,\{V\}_l\right)^2 dt}{\|\{V\}_l\|^2} \tag{3.21}$$

sous les conditions :

$\|\{V\}_l\| = 1$ et $\lambda_l \geq \lambda_{l+1}$.

Remarque :

Avec l'introduction du temps global t', l'équation (3.21) devient :

$$\lambda_l^{(n+1)} = \frac{\int_0^{(m-1)T+t_{j+1}} \left(\{a\}^{(n+1/2)T}(t') \,.\, \{V\}_l \right)^2 dt'}{\|\{V\}_l\|^2} \tag{3.22}$$

Les vecteurs $(\{V\}_l)_{l=1...s+1}$ sont les vecteurs propres de la matrice de covariance $[C]^{(n+1/2)}$ telle que :

$$[C]^{(n+1/2)} = \sum_{\beta=1}^{\beta=\alpha-1} \int_0^T \{a\}^{(n+1/2)}(t;\{p\}_\beta) \,.\, \{a\}^{(n+1/2)T}(t;\{p\}_\beta)\, dt + \int_0^{t_{j+1}} \{a\}^{(n+1/2)}(t;\{p\}_\alpha) \,.\, \{a\}^{(n+1/2)T}(t;\{p\}_\alpha)\, dt \tag{3.23}$$

Les évènements les plus significatifs sont sélectionnés en utilisant les critères suivants :

$$[V] = [\{V\}_1, ..., \{V\}_{\widetilde{s}}] \quad \text{avec } \lambda_{\widetilde{s}} > \varepsilon_{POD}\, \lambda_1 \quad \text{et } \lambda_{\widetilde{s}+1} \leq \varepsilon_{POD}\, \lambda_1 \tag{3.24}$$

où $\widetilde{s}$ est la nouvelle taille du modèle d'ordre réduit.
Finalement, le nouveau modèle d'ordre réduit est une réduction POD du modèle d'ordre réduit précédent :

$$\underline{\boldsymbol{\psi}}_l^{(n+1)} = \sum_{k=1}^{k=s+1} \underline{\boldsymbol{\psi}}_k^{(n+1/2)}\, V_{kl} \quad \forall l = 1...\widetilde{s} \tag{3.25}$$

$$a_l^{(n+1)}(t;\{p\}_\alpha) = \sum_{k=1}^{k=s+1} a_k^{(n+1/2)}(t;\{p\}_\alpha)\, V_{kl} \quad \forall l = 1...\widetilde{s} \tag{3.26}$$

3.3 Le concept d'évènements récurrents

Nous pouvons définir un évènement comme le couple formé par un vecteur de la base réduite et la variable d'état réduite associée. Un évènement est donc un couple $\left(\left\{ A_k^{(n)} \right\}, a_k^{(n)}(t') \right)$.

3.3.1 Définitions formelles

3.3.1.1 Évènement parfaitement récurrent

Un évènement $(\left\{ A_k^{(n)} \right\}, a_k^{(n)}(t'))$ est dit parfaitement récurrent si et seulement si :

$$\int_{t'_{j-1}}^{t'_j} (a_k^{(n)}(t'))^2 dt' = \int_0^T (a_k^{(n)}(t'))^2 dt' = E_{a_k} \qquad \forall j \in \{1, ..., m\} \tag{3.27}$$

où E_{a_k} est une constante indépendante de m.

Exemple

Un évènement périodique de période T, ou un évènement décalé dans le temps, sont des exemples d'évènement parfaitement récurrent.

3.3.1.2 Évènement récurrent

Définition

On mesure la contribution de $\left\{A_k^{(n)}\right\}$ (la $k^{\text{ème}}$ colonne de la matrice $[A]$) pour la simulation $S^{(j)}{}_{,j\in\{1,\ldots,m\}}$ à l'aide du facteur $\mu_{kj}^{(m)}$ défini comme tel :

$$\mu_{kj} = \frac{\int_{t'_{j-1}}^{t'_j} (\{q(t')\}^T \left\{A_k^{(n)}\right\})^2 dt'}{\left\{A_k^{(n)}\right\}^T \left\{A_k^{(n)}\right\}} \tag{3.28}$$

Afin de simplifier les notations, nous choisissons de ne pas écrire les indices m sur μ et λ, mais l'on garde en tête que m est fonction de n.

Propriété 1

$$\mu_{kj} = \int_{t'_{j-1}}^{t'_j} (a_k^{(n)}(t'))^2 dt' \tag{3.29}$$

Démonstration 1

Par construction, les fonctions propres sont orthogonales mais, pour des raisons pratiques, elles sont généralement choisies comme orthonormales. On a donc :

$$\left\{A_k^{(m)}\right\}^T \left\{A_i^{(m)}\right\} = \delta_{ki} \tag{3.30}$$

D'après (1.29) et cette propriété d'orthonormalité, on a :

$$\{q(t')\}^T \left\{A_k^{(n)}\right\} = a_k^{(n)}(t') \tag{3.31}$$

$$\left\{A_k^{(n)}\right\}^T \left\{A_k^{(n)}\right\} = 1 \tag{3.32}$$

Propriété 2

$$\lambda_k = \sum_{j=1}^{m} \mu_{kj} \tag{3.33}$$

Démonstration 2

La matrice $\left[A^{(n)}\right]$ est construite telle que la projection des colonnes de $\left[A^{(n)}\right]$ sur l'ensemble des $\left\{\tilde{q}(\tilde{t})\right\}$ soit maximum, c'est-à-dire que l'on cherche à maximiser la projection :

$$\lambda_k = \frac{\int_0^{mT} (\{\tilde{q}(t')\}^T \left\{A_k^{(n)}\right\})^2 dt'}{\left\{A_k^{(n)}\right\}^T \left\{A_k^{(n)}\right\}} \tag{3.34}$$

$$= \sum_{j=1}^{m} \frac{\int_{t'_{j-1}}^{t'_j} (\{\tilde{q}(t')\}^T \left\{A_k^{(n)}\right\})^2 dt'}{\left\{A_k^{(n)}\right\}^T \left\{A_k^{(n)}\right\}} \tag{3.35}$$

$$= \sum_{j=1}^{m} \mu_{kj} \tag{3.36}$$

Propriété 3

Des propriétés (3.29) et (3.33) découle la propriété suivante :

$$\lambda_k^{(m)} = \int_0^{mT} (\tilde{a}_k^{(m)}(\tilde{t}))^2 d\tilde{t} \tag{3.37}$$

Remarque

Nous avons précisé lors de l'introduction qu'il y a deux façons pour un évènement d'être significatif : être très intense sur une durée limitée ou l'être moins sur une durée plus longue. Nous illustrons ces deux cas en figure FIG. 3.3.

En effet, d'après (3.37), on a :

$$\lambda_i = \int_0^T (a_i^{(n)}(t'))^2 dt = \max_{t \in [0,T]} a_i^{(n)}(t')(t'_2 - t'_1) \tag{3.38}$$

Alors $\exists\, \tilde{a}_i^{(\tilde{n})}(t')$ tel que :

$$\tilde{\lambda}_i = \max_{t' \in [0,T]} \tilde{a}_i^{(\tilde{n})}(\tilde{t}'_2 - \tilde{t}'_1) = \lambda_i \tag{3.39}$$

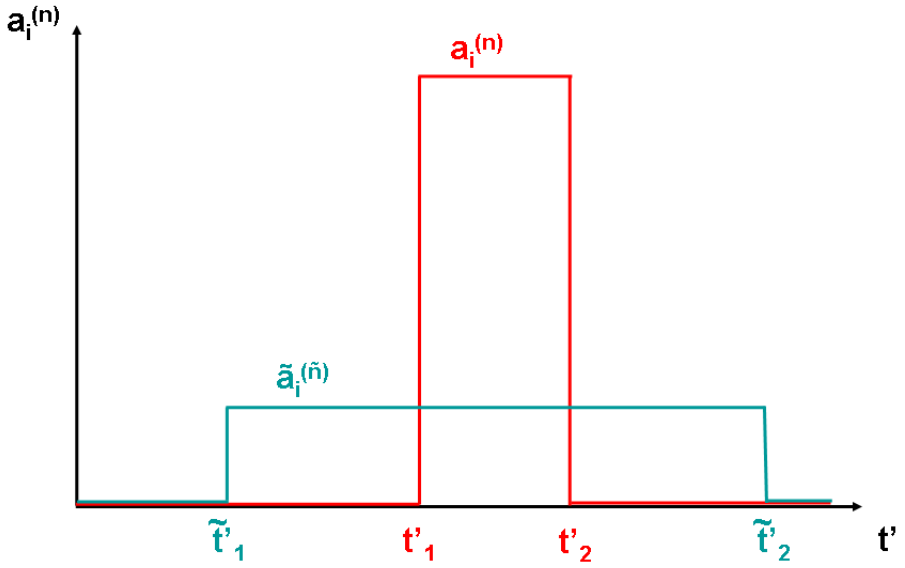


FIG. 3.3 – Différentes façons d'être significatif pour un évènement

En effet, il suffit de choisir judicieusement :

$$\max_{t' \in [0,T]} \tilde{a}_i^{(\tilde{n})} = \frac{\max\limits_{t \in [0,T]} a_i^{(n)}(t')}{(\tilde{t}'_2 - \tilde{t}'_1)}(t'_2 - t'_1) \tag{3.40}$$

Définition

On mesure la récurrence de l'évènement $(\left\{A_k^{(n)}\right\}, a_k^{(n)}(t'))$ grâce au facteur I_k suivant :

$$I_k = \sqrt[m]{\prod_{j=1}^{m} \mu_{kj}} \tag{3.41}$$

Définition

L'évènement $(\left\{A_k^{(n)}\right\}, a_k^{(n)}(t'))$ est dit récurrent si et seulement si :

$$\frac{mI_k}{\lambda_1} > \varepsilon_{POD} \tag{3.42}$$

où ε_{POD} est le paramètre seuil défini en 1.43 (et fixé par l'utilisateur, par exemple, à 10^{-8}).

Propriété

Si l'évènement le plus significatif est un évènement parfaitement récurrent, on a :

$$I_1 = \mu_{11}\lambda_1 = m\mu_{11}\frac{mI_1}{\lambda_1} = 1 \tag{3.43}$$

3.3.1.3 Évènement parfaitement spécifique

Rappelons qu'un évènement $(\left\{A_k^{(n)}\right\}, a_k^{(n)}(t'))$ est significatif si $\lambda_k > \varepsilon_{POD}\lambda_1$.

Définition

Un évènement $(\left\{A_k^{(n)}\right\}, a_k^{(n)}(t'))$ est dit parfaitement spécifique si et seulement si :

$$\exists j_0 \text{ tel que } \mu_{kj} = 0 \quad \forall j \neq j_0 \tag{3.44}$$

Propriété

$$\lambda_k = \mu_{kj_0} = \tilde{\mu}_{kj_0} \tag{3.45}$$

où μ_{kj_0} est une constante indépendante du nombre de simulations contenues dans la suite de simulations.

Cette propriété découle de la propriété (3.33) et de la définition ci-dessus (3.44).

3.3.2 Propriété de dégradation du modèle d'ordre réduit

Soit $\left\{q^{(n)}\right\}(t') = \sum_{k=1}^{s} \left\{A_k^{(n)}\right\} a_k^{(n)}(t')$ tel que :

$\exists r \in \{1, ..., s\}$ tel que l'évènement $(\left\{A_r^{(n)}\right\}, a_r^{(n)}(t'))$ soit parfaitement récurrent $\forall m$.

$\exists z \neq r \in \{1, ..., s\}$ tel que l'évènement $(\left\{A_z^{(n)}\right\}, a_z^{(n)}(t'))$ soit parfaitement spécifique $\forall m$.

Alors il existe un nombre de simulations m tel que :

$$\forall m > m_0, \frac{\lambda_z}{\lambda_1} < \varepsilon_{POD} \tag{3.46}$$

C'est à dire tel que l'évènement parfaitement spécifique devienne négligeable au sens de la POD.
Démonstration

$$\lambda_z = \tilde{\mu}_{zj_0} \tag{3.47}$$
$$\lambda_r = mE_{a_r} \tag{3.48}$$
$$\tag{3.49}$$

Ceci implique :

$$\frac{\lambda_z}{\lambda_r} = \frac{C}{m} \tag{3.50}$$

avec $C = \frac{\tilde{\mu}_{zj_0}}{E_{a_r}}$ constant.

Donc

$$\exists\, m_0 \text{ tel que } \frac{\lambda_z}{\lambda_r} < \varepsilon_{POD} \tag{3.51}$$

Or, comme on a toujours $\lambda_{max} = \lambda_1$, on a aussi :

$$\frac{\lambda_z}{\lambda_r} > \frac{\lambda_z}{\lambda_1} \tag{3.52}$$

Donc :

$$\frac{\lambda_z}{\lambda_r} < \varepsilon_{POD} \Rightarrow \frac{\lambda_z}{\lambda_1} < \varepsilon_{POD} \tag{3.53}$$

D'après le critère de sélection d'évènements significatifs, cela signifie que l'évènement qui était parfaitement spécifique devient négligeable, et n'appartient plus au modèle d'ordre réduit. Or, si cet évènement spécifique doit intervenir à la fin de la suite de simulations ($j_0 = m$) afin d'aider la convergence d'un processus d'optimisation, ceci peut compromettre le succès de l'optimisation.

3.3.3 Mise en évidence de la présence d'évènements récurrents sur un exemple numérique

3.3.3.1 Objectifs

Nous allons mettre en évidence la présence d'évènements récurrents sur un modèle linéaire d'une barre bi-encastrée. L'objectif est de montrer l'existence d'évènements récurrents lors de l'étude d'une suite de calculs de sensibilité par Différences Finies en utilisant une méthode de réduction de modèle.

3.3.3.2 Description du modèle

Le modèle que nous avons choisi est un problème linéaire de traction d'une barre bi-encastrée avec un chargement évoluant sur les 3 derniers noeuds (FIG.3.4). La longueur de la barre est $L = 2$, cette barre est maillée en $n = 600$ noeuds.

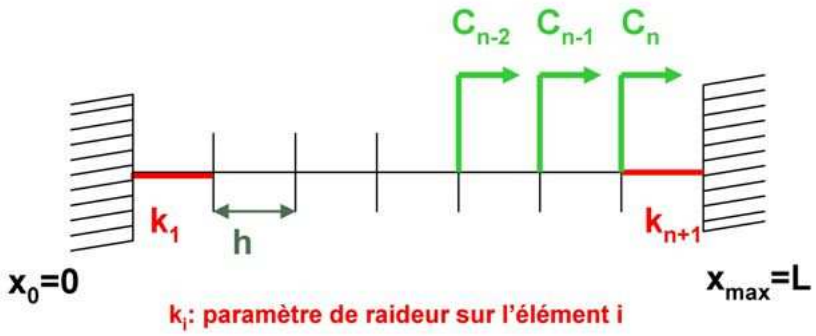


FIG. 3.4 – Barre bi-encastrée.

Nous considérons un chargement évoluant sur les 3 derniers noeuds, donc on aura :

$$\underline{\boldsymbol{F}} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ C_{n-2}(t) \\ C_{n-1}(t) \\ C_n(t) \end{pmatrix} \tag{3.54}$$

Dans notre cas, nous choisirons $C_{n-2}(t) = C_{n-1}(t) = C_n(t) = 1$.

Considérons une suite de calculs, avec un modèle évoluant au cours de cette suite. Un calcul de l'évolution de l'état d'un système, et un calcul de sensibilité en base réduite sont faits, ce qui donne une variation significative ou infinitésimale des paramètres.
Nous présenterons une suite de $m = 50$ simulations, puis une suite de $m = 100$ simulations.
Dans ce cas, la matrice des réponses est :

$$\left[\tilde{Q}^{(j)}\right] = \left[\left\{\tilde{q}_1^{(j)}\right\}, \left\{\tilde{q}_2^{(j)}\right\}, \left\{\tilde{q}_3^{(j)}\right\}\right] \tag{3.55}$$

La suite de simulations obtenue est représentée sur la figure FIG. 3.5.

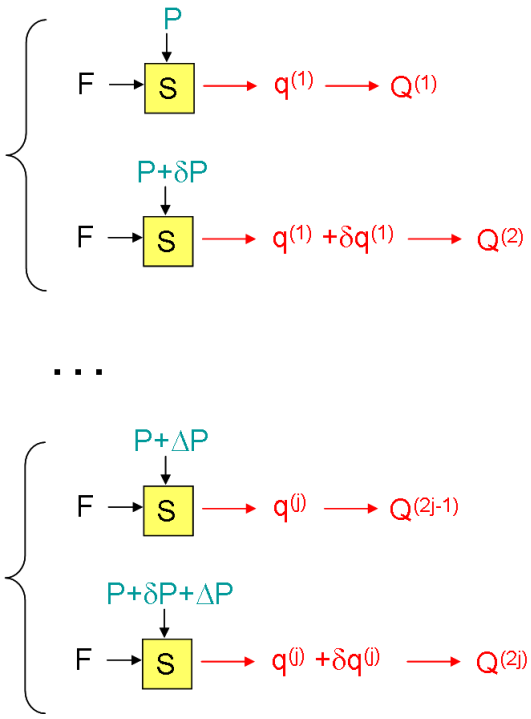


FIG. 3.5 – Suite de simulations traitée

Les questions que nous nous posons, et auxquelles nous espérons trouver une réponse sur cet exemple, sont les suivantes :

- Existe-t-il des évènements récurrents ?
- Quelle est la conséquence de la présence de ces évènements récurrents sur la précision du calcul de sensibilité en base réduite ? (la base réduite étant la base POD définie pour représenter l'ensemble des calculs).
- Quelle importance ont-ils ?

3.3.3.3 Algorithme de réduction de modèle

Soit $[P]$ une matrice constituée des m vecteurs colonnes $\{p\}_{\alpha,\alpha=1,\dots,m} \in \mathbb{R}^{n+1}$ constitués des paramètres de raideurs $k_i^{(\alpha)}{}_{,i=1,\dots,n+1}$ (ces paramètres sont choisis aléatoirement).
ξ est un paramètre de perturbation du vecteur de paramètres. Dans nos exemples, il sera d'abord de $0.9k_0$, puis de $0.3k_0$, où $k_0 = 0.2$.
Enfin, nous choisissons un paramètre de perturbation de l'état du système $\varepsilon_{dp} = 0.1n$.
Le vecteur des perturbations associé est un vecteur de taille $n+1$ tel que :

$$\{\delta p\} = \begin{pmatrix} \varepsilon_{dp} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

. La suite de calculs est effectuée en base réduite, et les paramètres de la méthode de réduction de modèle sont $\epsilon_R = 10^{-3}$ et $\varepsilon_{POD} = 10^{-8}$.

La sensibilité aux variations de paramètres de la manière suivante :

$$\{S_q\} = \frac{\{\tilde{q}\} - \{q\}}{||\{\delta p\}||} \tag{3.62}$$

La sensibilité calculée en base réduite est déterminée de la sorte :

$$\{S_{Aa}\} = \frac{\{A\}_k \cdot a - \{A\}_k \cdot \tilde{a}}{||\{\delta p\}||} \tag{3.63}$$

3.3.3.4 Résultats et analyse

Nous voulons prouver que la précision du calcul doit se détériorer avec le nombre de simulations, donc l'erreur sur la sensibilité doit augmenter avec le nombre de simulations. Cette erreur est calculée de la manière suivante :

$$E_s = 100 \cdot \frac{||\{S_q\} - \{S_{Aa}\}||}{||\{S_q\}||} \tag{3.64}$$

où $\{S_q\}$ et $\{S_{Aa}\}$ sont définis respectivement en (3.62) et (3.63). C'est effectivement ce que l'on constate sur la figure FIG. 3.6 à partir de 15 simulations effectuées.

Nous pouvons maintenant vérifier si la présence d'évènements récurrents est d'autant plus importante au fur et à mesure des simulations.

for $\alpha = 1$ **to** m **do**

1. Définition des paramètres de raideur sur chaque élément

$$\{k\}_i = \xi \{p\}_i + \{k_0\} \tag{3.56}$$

2. Construction de la matrice de rigidité

$$[K] = \begin{bmatrix} k_1^{(i)} + k_2^{(i)} & -k_2^{(i)} & 0 & . & . & . & 0 \\ -k_2^{(i)} & k_2^{(i)} + k_3^{(i)} & -k_3^{(i)} & . & . & . & . \\ 0 & -k_3^{(i)} & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & -k_n^{(i)} \\ 0 & . & . & . & 0 & -k_n^{(i)} & k_n^{(i)} + k_{n+1}^{(i)} \end{bmatrix} \tag{3.57}$$

3. Résolution du modèle détaillé $[K]\,\underline{\boldsymbol{q}} = \underline{\boldsymbol{F}}$;
Chaque chargement obtenu $\underline{\boldsymbol{q}}_{ii=1,\ldots,3}$ est stocké dans la matrice des snapshots

$$[Q]^{(i)} = [[Q]^{(i-1)}, q_1, q_2, q_3] \tag{3.58}$$

4. Définition des nouveaux paramètres de raideur

$$\left\{\tilde{k}\right\}_i = \xi \{p\}_i + \{dp\} + \{k0\} \tag{3.59}$$

5. Construction de la nouvelle matrice de rigidité $\left[\tilde{K}\right]$;

6. Résolution d'un nouveau modèle détaillé $\left[\tilde{K}\right]\underline{\tilde{\boldsymbol{q}}} = \underline{\boldsymbol{F}}$.;

7. Sélection des s valeurs propres de $[Q]^T[Q]$ qui sont supérieures à ε_{POD} ainsi que de la matrice des vecteurs propres correspondants $[B]$;

8. Détermination de la matrice de réduction de base $[A] = [Q]\,[B]$;

9. Calcul des variables d'état réduites $\underline{\boldsymbol{a}}_{kk=1,\ldots,s}$ telles que

$$\underline{\boldsymbol{A}}_k^T\,[K]\,\underline{\boldsymbol{A}}_k \cdot a_k = \underline{\boldsymbol{A}}_k^T \underline{\boldsymbol{F}} \tag{3.60}$$

10. Calcul des nouvelles variables d'état réduites $\underline{\tilde{\boldsymbol{a}}}_{kk=1,\ldots,s}$ telles que

$$\underline{\boldsymbol{A}}_k^T \left[\tilde{K}\right] \underline{\boldsymbol{A}}_k \cdot \tilde{a}_k = \underline{\boldsymbol{A}}_k^T \underline{\boldsymbol{F}} \tag{3.61}$$

end

Algorithm 8: Algorithme de calcul de sensibilité en base réduite.

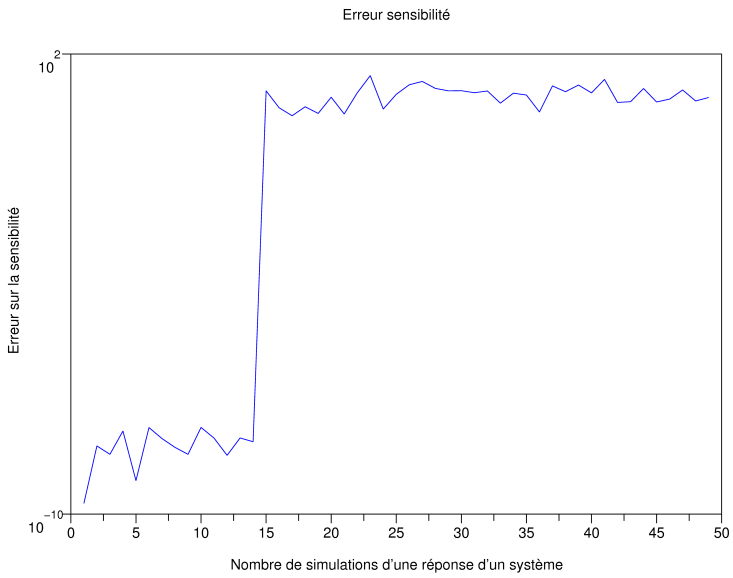


FIG. 3.6 – Erreur sensibilité E_s grandissante avec le nombre de simulations

Nous mesurons les évènements récurrents en calculant le facteur d'intensité F_k de la manière suivante :

$$F_k^{(m)} = \sum_k \lambda_k f_k^{(m)} \tag{3.65}$$

où

$$f_k^{(m)} = \begin{cases} 1 & \text{si } \frac{mI_k}{\lambda_1} > \varepsilon_{POD} \\ 0 & \text{sinon.} \end{cases} \tag{3.66}$$

Nous avons représenté sur la courbe de gauche de la figure 3.7 l'intensité des évènements récurrents F_k en fonction du nombre de simulations pour une suite de 50 calculs, et sur la courbe de droite, la même courbe pour une suite de 100 calculs.

Et nous pouvons effectivement observer que, plus on effectue de calculs dans une suite, plus l'intensité des évènements récurrents augmente (on passe d'une intensité de $5,5.10^6$ pour une suite de 50 calculs à une intensité de $1,1.10^7$ pour une suite de 100 calculs.)

Il reste à vérifier que l'intensité des évènements récurrents doit augmenter en diminuant la perturbation des paramètres.

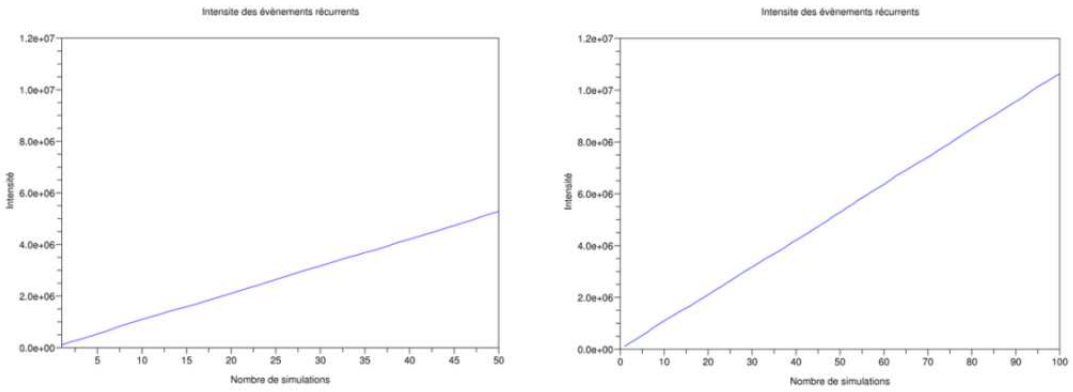


FIG. 3.7 – Intensité des ER F_k pour des suites de 50 et 100 calculs

En effet, plus la perturbation est faible, plus on va reproduire des calculs similaires, et donc plus l'effet des évènements récurrents se fera sentir.

Ainsi, en prenant sur la courbe droite de la figure FIG. 3.8 une perturbation 3 fois plus faible que celle sur la courbe de gauche, on observe que l'intensité des évènements récurrents a augmenté : on passe d'une intensité de $3.5.10^6$ à une intensité de $5,3.10^6$ pour une perturbation 3 fois plus faible.

Cela prouve que la présence des évènements récurrents se fait bien sentir sur cet exemple.

Une stratégie proposée pour gérer ce problème est d'oublier une partie des évènements antérieurs : une nouvelle suite de calculs est alors traitée en oubliant quasiment tous les calculs antérieurs et en ne prenant en compte plus que les deux derniers calculs effectués.
On compare en figure FIG. 3.9 l'erreur obtenue avec tous les calculs pris en compte (FIG. 3.6) avec celle obtenue en ne prenant en compte plus que les 2 derniers calculs effectués.
L'erreur obtenue en oubliant une partie des calculs antérieurs est devenue complètement négligeable.

3.3.3.5 Constat

Pour conclure cette partie, rappelons le principe de la suite de calculs : celui-ci réside dans le fait que les connaissances acquises au cours d'une suite

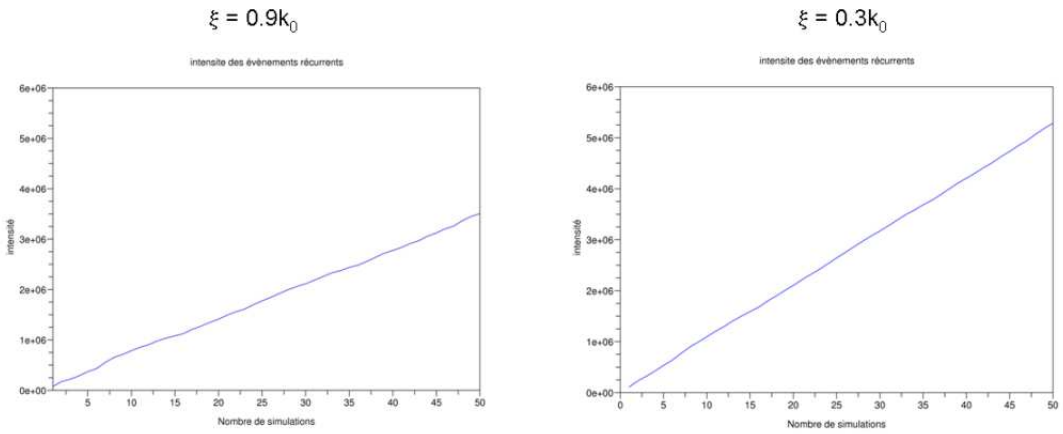


FIG. 3.8 – Intensité des ER F_k pour une perturbation plus ou moins faible

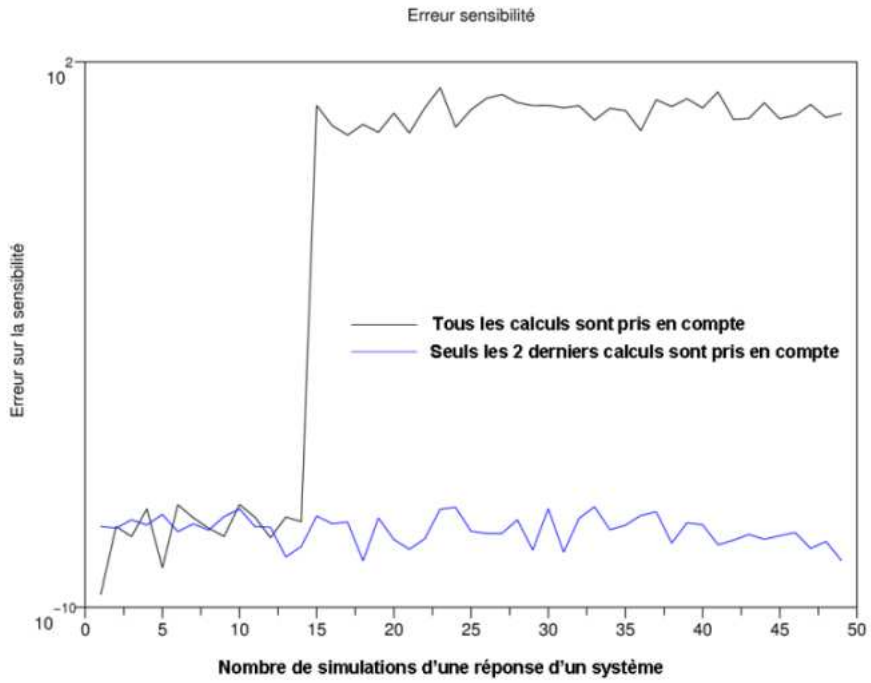


FIG. 3.9 – Calculs antérieurs oubliés

de simulations doivent permettre de faciliter les calculs en cours.
Mais, lorsqu'une base est construite pour essayer de représenter les évènements significatifs contenus dans l'ensemble des simulations, cela induit une perte de représentation des réponses aux calculs de sensibilité.
Alors l'accumulation des résultats conduit à une dégradation des connaissances acquises, à cause des évènements récurrents : les réponses aux calculs de sensibilité sont en effet de moins en moins significatifs face à des évènements présents dans chacune des simulations.

3.3.4 Atténuation des évènements récurrents

Le problème des évènements récurrents a été mis en évidence sur un exemple. Nous proposons de développer une méthode de réduction adaptative avec atténuation de ces évènements récurrents, afin d'obtenir un traitement efficace d'une suite de simulations. Ainsi, le modèle d'ordre réduit construit par adaptations successives devrait permettre de mieux représenter les dernières solutions de la suite de simulations, que les premières.
Pour cela, il faut modifier la construction de la base pour effectuer une suite de calculs donnant un résultat satisfaisant pour la dernière simulation en cours.

Le principe de cette solution est de savoir oublier pour mieux adapter : une stratégie d'oubli des évènements récurrents est proposée, en introduisant un coefficient multiplicateur $0 \leq \gamma \leq 1$ dans la définition de λ_k (la somme provenant de l'accumulation de simulations) :

$$\lambda_k = \sum_{j=1}^{m} \mu_{kj} \text{ devient } \lambda_k = \sum_{j=1}^{m} \gamma^{m-j} \mu_{kj} \tag{3.67}$$

Ce facteur, également appelé "facteur d'oubli" va nous permettre de choisir entre :

- un modèle d'ordre réduit qui décrit tous les états ($\gamma = 1$ correspond à l'approche standard sans atténuation des évènements récurrents)
- un modèle d'ordre réduit plus précis pour les dernières valeurs de paramètres que pour les premières ($\gamma < 1$)

L'objectif de nos travaux étant d'étudier les évènements récurrents afin de maîtriser la qualité des calculs en base réduite en cours de processus d'optimisation, un démonstrateur a été développé dans le code ZéBuLoN. Nous pourrons ainsi évaluer l'apport de cette méthode pour gérer la présence des évènements récurrents.

3.4 Algorithme APHR avec atténuation des évènements récurrents

L'algorithme APHR pour une suite de simulations présenté en section 3.2 est modifié afin de prendre en compte l'atténuation des évènements récurrents. Nous présentons ici l'algorithme entier, mais en fin de compte, seules les équations (3.21) et (3.23) ont été modifiées.

Étape 1 :
Estimation des variables détat réduites.

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_{\alpha}) = \sum_{k=1}^{k=\widehat{m}} \underline{\boldsymbol{\psi}}_{k}^{(n)}(\underline{\boldsymbol{X}}\,)\,a_{k}^{(n)}(t;\{p\}_{\alpha}) + \underline{\boldsymbol{u}}_{ch}(\underline{\boldsymbol{X}}\,,t) \quad (3.68)$$

$$\int_{\Omega_{\Pi}} \underset{\approx}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^{*}) \,:\, \underset{\approx}{\boldsymbol{\Sigma}}(\underset{\approx}{\boldsymbol{\varepsilon}}_{\tau}(\underline{\boldsymbol{u}})\,,\,\tau \leq t;\{p\}_{\alpha})\; d\Omega \;-\int_{\partial_{f}\Omega_{\Pi}} \underline{\boldsymbol{u}}^{*}\,.\,\underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_{\alpha})\; d\Gamma = \int_{\Omega_{\Pi}} \underline{\boldsymbol{u}}^{*}.\underline{\boldsymbol{r}}_{v} d\Omega + \int_{\partial_{f}\Omega_{\Pi}} \underline{\boldsymbol{u}}^{*}.\underline{\boldsymbol{r}}_{b} d\Gamma$$
$$\forall\, \underline{\boldsymbol{u}}^{*} \in \mathcal{V}_{\Pi}^{(n)},\;\; \forall t \in\,]t_{0},t_{f}] \quad (3.69)$$

Étape 2 :
Extrapolation des variables internes de Ω_{Π} à $\overline{\Omega}_{\Pi}$.

$$\{b\}^{(n)}\,(t_{i+1};\{p\}_{\alpha}) = arg\,min_{\{y\}}\,H(\{y\}) \quad (3.70)$$

$$H(\{y\}) = \int_{\Omega_{\Pi}} < \mathbf{z}_{ROM} - \sum_{k=1}^{k=\xi} \mathbf{Y}_{k}^{(n)}(\underline{\boldsymbol{X}}\,)\,y_{k}\;,\; \mathbf{z}_{ROM} - \sum_{k=1}^{k=\xi} \mathbf{Y}_{k}^{(n)}(\underline{\boldsymbol{X}}\,)\,y_{k} > \; d\Omega \quad (3.71)$$

$$\mathbf{z}_{ROM}(\underline{\boldsymbol{X}}\,,t_{i+1};\{p\}_{\alpha}) = \sum_{k=1}^{k=\xi} \mathbf{Y}_{k}^{(n)}(\underline{\boldsymbol{X}}\,)b_{k}^{(n)}(t_{i+1};\{p\}_{\alpha}) \quad \forall \underline{\boldsymbol{X}} \in \overline{\Omega}_{\Pi} \quad (3.72)$$

Étape 3 :
Évaluation de l'indicateur d'erreur η_{POD} relatif au résidu tronqué $\underline{\tilde{\boldsymbol{\mathcal{R}}}}$:

$$\underline{\tilde{\boldsymbol{\mathcal{R}}}}_{j} \;=\; \int_{\Omega_{\Pi}} \underset{\approx}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^{*}) \,:\, \underset{\approx}{\boldsymbol{\Sigma}}(\underset{\approx}{\boldsymbol{\varepsilon}}_{\tau}(\underline{\boldsymbol{u}})\,,\,\tau \leq t;\{p\}_{\alpha})\; d\Omega \;-\int_{\partial_{f}\Omega_{\Pi}} \underline{\boldsymbol{u}}^{*}\,.\,\underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_{\alpha})\; d\Gamma \quad (3.73)$$

avec

$$\begin{aligned} \underline{\boldsymbol{u}}^{*} &= \underline{\boldsymbol{N}}\,(\underline{\boldsymbol{X}}\,)(\max_{i}\Pi_{ij}) \\ \eta_{POD} &= \|\underline{\tilde{\boldsymbol{\mathcal{R}}}}\,\| \end{aligned}$$

Étape 4 :

- Si $\eta_{POD} < \epsilon_R\|f\|$, alors $\quad \delta\underline{\boldsymbol{u}}_h(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) = 0,$
- Sinon, la correction des déplacements est telle que :

$$\begin{aligned} &\int_\Omega \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^{*}) + \delta\underline{\boldsymbol{u}}_h^{(n)}) \;:\; \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{\varepsilon}}_\tau(\underline{\boldsymbol{u}}_{POD}^{(n)} + \delta\underline{\boldsymbol{u}}_h^{(n)})\,,\, \tau \le t; \{p\}_\alpha)\;\; d\Omega \\ &-\int_{\partial_f\Omega} \underline{\boldsymbol{u}}^{*}\,.\,\underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_\alpha)\;\; d\Gamma = \int_\Omega \underline{\boldsymbol{u}}^{*}.\underline{\boldsymbol{r}}_v d\Omega + \int_{\partial_f\Omega} \underline{\boldsymbol{u}}^{*}.\underline{\boldsymbol{r}}_b d\Gamma \\ &\qquad\qquad \forall\, \underline{\boldsymbol{u}}^{*} \in \mathcal{V}_h,\;\; \forall t \in\,]t_0, t_f]\,, \quad \forall\, \underline{\boldsymbol{X}} \in \partial_U\Omega \end{aligned} \tag{3.74}$$

Étape 5 :

$$\delta_\perp\underline{\boldsymbol{u}}_h^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) = \delta\underline{\boldsymbol{u}}_h^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) - \widehat{\delta\underline{\boldsymbol{u}}_h}^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) \tag{3.75}$$

$$\widehat{\delta\underline{\boldsymbol{u}}_h}^{(n)}(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}\,)\, \delta a_k^{(n)}(t_{i+1}; \{p\}_\alpha) \tag{3.76}$$

$$\{\delta a\}^{(n)}\,(t_{i+1}; \{p\}_\alpha) = \arg\min_{\{y\}} \int_\Omega \left\| \delta\underline{\boldsymbol{u}}_h(\underline{\boldsymbol{X}}\,, t_{i+1}; \{p\}_\alpha) - \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}\,)\, y_k \right\|^2 d\Omega \tag{3.77}$$

Alors, la nouvelle base étendue $\left(\underline{\boldsymbol{\psi}}_k^{(n+1/2)}\right)_{k=1\ldots s+1}$ est construite en conservant les prévisions précédentes telles que :

$$\underline{\boldsymbol{\psi}}_k^{(n+1/2)} = \underline{\boldsymbol{\psi}}_k^{(n)} \quad k \le s \tag{3.78}$$

$$\underline{\boldsymbol{\psi}}_{s+1}^{(n+1/2)} = \frac{1}{\|\delta_\perp\underline{\boldsymbol{u}}_h\|}\, \delta_\perp\underline{\boldsymbol{u}}_h \tag{3.79}$$

$$a_k^{(n+1/2)}(\tau; \{p\}_\alpha) = a_k^{(n)}(\tau; \{p\}_\alpha) \quad \tau \le t_{i+1} \quad k \le s \tag{3.80}$$

$$a_k^{(n+1/2)}(\tau; \{p\}_\beta) = a_k^{(n)}(\tau; \{p\}_\beta) \quad \tau \le T \quad k \le s \quad \beta < \alpha \tag{3.81}$$

$$a_{s+1}^{(n+1/2)}(\tau; \{p\}_\beta) = 0 \quad \tau \le T \quad \beta < \alpha \tag{3.82}$$

$$a_{s+1}^{(n+1/2)}(\tau; \{p\}_\alpha) = 0 \quad \tau \le t_i \tag{3.83}$$

$$a_{s+1}^{(n+1/2)}(t_{i+1}; \{p\}_\alpha) = \|\delta_\perp\underline{\boldsymbol{u}}_h\| \tag{3.84}$$

On effectue une décomposition POD sur les variables réduites afin d'éviter une évolution constante de la taille du modèle d'ordre réduit. Cette POD consiste à trouver les vecteurs $(\{V\}_l)_{l=1...s+1}$ qui maximisent la projection sur les variables d'état réduites suivante. Le facteur d'oubli γ permettant d'atténuer l'effet des évènements récurrents est introduit dans la formulation de $\lambda_l^{(n+1)}$ tel que :

$$\lambda_l^{(n+1)} = \frac{\sum_{\beta=1}^{\beta=\alpha-1} \gamma^{\alpha-1-\beta} \int_0^T \left(\{a\}^{(n+1/2)T}(t;\{p\}_\beta) \,.\, \{V\}_l \right)^2 dt}{\|\{V\}_l\|^2} + \frac{\int_0^{t_{j+1}} \left(\{a\}^{(n+1/2)T}(t;\{p\}_\alpha) \,.\, \{V\}_l \right)^2 dt}{\|\{V\}_l\|^2} \tag{3.85}$$

sous les conditions :

$\|\{V\}_l\| = 1$ et $\lambda_l \geq \lambda_{l+1}$.

Notons bien que dans l'équation (3.85), γ est élevé à la puissance $\alpha - 1 - \beta$. Il ne s'agit pas d'un indice.

Les vecteurs $(\{V\}_l)_{l=1...s+1}$ sont les vecteurs propres de la matrice de covariance $[C]^{(n+1/2)}$ telle que :

$$[C]^{(n+1/2)} = \sum_{\beta=1}^{\beta=\alpha-1} \gamma^{\alpha-1-\beta} \int_0^T \{a\}^{(n+1/2)}(t;\{p\}_\beta) \,.\, \{a\}^{(n+1/2)T}(t;\{p\}_\beta)\, dt + \int_0^{t_{j+1}} \{a\}^{(n+1/2)}(t;\{p\}_\alpha) \,.\, \{a\}^{(n+1/2)T}(t;\{p\}_\alpha)\, dt \tag{3.86}$$

Les évènements les plus significatifs sont sélectionnés en utilisant les critères suivants :

$$[V] = [\{V\}_1, ..., \{V\}_{\widetilde{s}}] \quad \text{avec } \lambda_{\widetilde{s}} > \varepsilon_{POD}\, \lambda_1 \quad \text{et } \lambda_{\widetilde{s}+1} \leq \varepsilon_{POD}\, \lambda_1 \tag{3.87}$$

où $\widetilde{s}$ est la nouvelle taille du modèle d'ordre réduit.
Finalement, le nouveau modèle d'ordre réduit est une réduction POD du

modèle d'ordre réduit précédent :

$$\underline{\boldsymbol{\psi}}_l^{(n+1)} = \sum_{k=1}^{k=s+1} \underline{\boldsymbol{\psi}}_k^{(n+1/2)} V_{kl} \quad \forall l = 1...\widetilde{s} \tag{3.88}$$

$$a_l^{(n+1)}(t; \{p\}_\alpha) = \sum_{k=1}^{k=s+1} a_k^{(n+1/2)}(t; \{p\}_\alpha) V_{kl} \quad \forall l = 1...\widetilde{s} \tag{3.89}$$

3.5 Implémentation de la méthode dans le code de calcul ZéBuLoN

Développé conjointement par l'ONERA, Northwest Numerics (Seattle, USA), et le Centre des Matériaux, ZéBuLoN est un code de calcul Éléments Finis adapté aux problèmes de mécanique non linéaire. Il est programmé en C++.

La méthode de réduction de modèles a été développée dans ce code de calcul, et le diagramme des classes de cette méthode se trouve en figure FIG. 3.10.

Les étapes d'adaptation du modèle de taille réduite sont implémentées dans la classe Reduced_state, qui hérite de la classe Reduced_Order_Model, comme l'on peut le voir en figure FIG. 3.10.

L'objet de notre travail est l'implémentation du facteur d'oubli. Celui-ci se nomme "gamma_history" dans le code ZéBuLoN. Sa valeur par défaut est déclarée dans la classe Reduced_Order_Model.

Ce facteur d'oubli vient multiplier la matrice de covariance $[M]$ utilisée pour la résolution du problème aux valeurs propres. L'implémentation du facteur d'oubli se fait donc au niveau de la fonction Eigenproblem() (problème aux valeurs propres), fonction de la classe Reduced_State. Le code C++ des fichiers Reduced_state.h et Reduced_state.c se trouvent respectivement en annexes 4.5 et 4.5.

Les lignes de codes modifiées pour le fichier Reduced_state.h sont les lignes 76 et 93, c'est-à-dire la déclaration des fonctions utilisant gamma_history : manage_recurrent_event() et init_cov_hist(). Les lignes de codes modifiées pour le fichier Reduced_state.c sont les lignes 362 à 371 et 860 à 869, soit l'implémentation de ces deux fonctions utilisant gamma_history.

Cette nouvelle approche est mise en oeuvre sur une suite de simulations et un problème inverse dans les sections suivantes. Nous allons traiter quelques exemples numériques (dont notre fil conducteur) en utilisant ce code ZéBuLoN modifié avec l'implémentation du facteur d'oubli. Nous

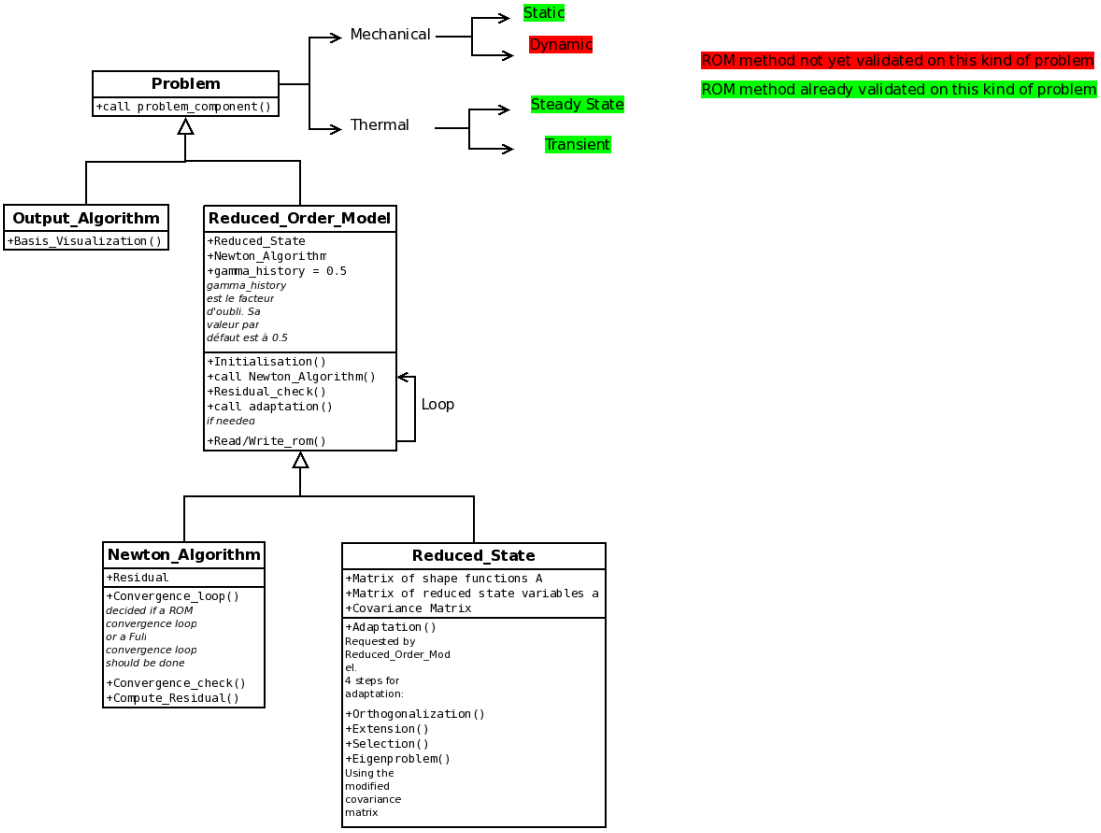


FIG. 3.10 – Diagramme des classes de la méthode de réduction de modèles dans le code de calcul ZéBuLoN.

allons évaluer l'apport de la méthode proposée pour gérer la présence des évènements récurrents. Dans un premier temps, nous allons évaluer l'effet des évènements récurrents sur une suite de simulations par une méthode de réduction de modèles, et vérifier si l'on peut confirmer l'apport de la solution proposée.

3.6 Étude d'une suite de simulations non-linéaires

3.6.1 Objectif

Il s'agit d'illustrer numériquement la propriété de dégradation d'un modèle d'ordre réduit en présence d'évènements récurrents.
Nous évaluerons l'effet de ces évènements récurrents sur une suite de simulations par une méthode de réduction de modèles, sur le logiciel ZéBuLoN.
Le cas d'étude est le modèle élastoplastique "fil conducteur".

3.6.2 Présentation de la suite de calculs

Soient deux types de simulations S_1 et S_2.
Nous allons comparer trois suites de calculs, celles présentées en figure FIG. 3.11.

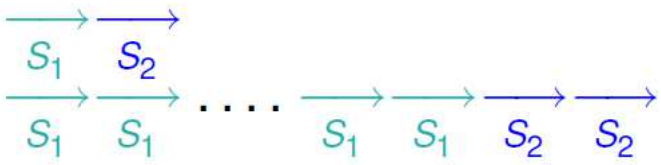


FIG. 3.11 – Différentes suites de simulations

Nous regarderons alors l'efficacité de l'approximation en base réduite de la dernière simulation S_2 pour chacune des suites de simulations.
L'efficacité sera mesurée par le nombre de calculs en base complète nécessaires pour obtenir une prévision de la qualité désirée, ou alors par la qualité de la prévision obtenue sans procédure d'adaptation.
Nous voudrons également savoir si la première simulation S_2 permet d'acquérir des connaissances suffisantes pour traiter efficacement la seconde simulation S_2.
La première simulation S_2 permet d'adapter le modèle d'ordre réduit à la

nouvelle valeur des paramètres.
La seconde simulation S_2 est réalisée avec une base figée afin d'évaluer l'effet des évènements récurrents sur la dernière adaptation du modèle d'ordre réduit.
Pour le cas d'étude "fil conducteur", soient $\varepsilon_p^{(ref)}$ et $\varepsilon_p^{(m)}$ les valeurs maximum de plasticité cumulée obtenues après les suites respectivement présentées en figure FIG. 3.12.

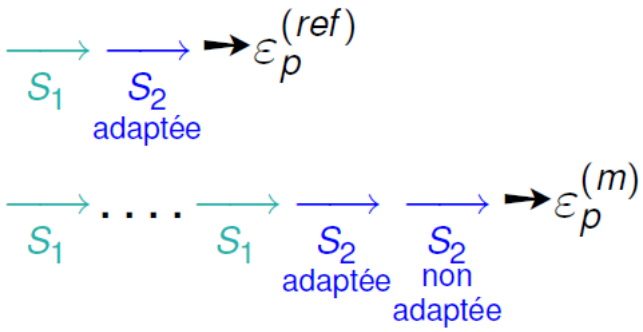


FIG. 3.12 – Obtention de la plasticité cumulée maximum

Nous pouvons alors calculer l'erreur sur la plasticité de la manière suivante :

$$E_p^{(m)} = 100 \left| \frac{\varepsilon_p^{(m)} - \varepsilon_p^{(ref)}}{\varepsilon_p^{(ref)}} \right| \tag{3.90}$$

La propriété de dégradation du modèle d'ordre réduit a été illustrée sur des suites de simulations de deux différents problèmes que nous allons présenter dans la section suivante.
Chacune de ces suites de simulations est constituée de m simulations réalisées dont les paramètres sont contenus dans le vecteur $\{p\} = \{p_1\}$ suivies de 2 simulations avec $\{p\} = \{p_2\}$.
Pour chaque cas étudié, nous préciserons les valeurs de $\{p_1\}$ et $\{p_2\}$, ainsi que la valeur de m.
Dans tous les cas, la différence entre les matériaux 1 et 2 sera la modification de la limite d'élasticité R_0. Les autres paramètres resteront inchangés d'une simulation à l'autre.
Ainsi, l'objectif de la simulation S_2 est donc d'évaluer l'influence d'une modification de la limite d'élasticité, et de savoir quelle sera l'efficacité du traitement de la connaissance évaluée par rapport à cet objectif.

Les résultats obtenus sont présentés et analysés dans la section suivante.

3.6.3 Présentation du cas d'étude

Afin d'illustrer la propriété de dégradation d'un modèle d'ordre réduit en présence d'évènements récurrents dans la simulation de problèmes de mécanique classiques, nous choisissons de travailler sur notre cas d'étude fil conducteur, dont nous rappelons le maillage et ses conditions aux limites ci-dessous (FIG. 3.13). Sur toute la partie supérieure de l'éprouvette, nous imposons une pression $P(t) = 14.72t$.

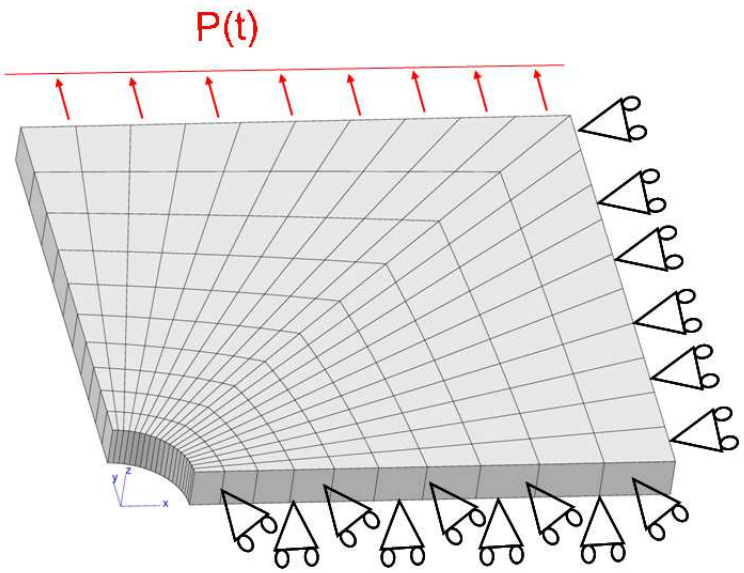


FIG. 3.13 – Maillage d'un huitième de l'éprouvette et conditions aux limites.

Les valeurs des paramètres du modèle sont données dans la table TAB. 3.1 suivante. Le seuil de plasticité R_0 a été divisé par 600 et passe ainsi de 60000 MPa pour la simulation S_1 à 100 MPa pour S_2, soit d'un matériau élastoplastique à un matériau totalement élastique.

La courbe représentant la loi de comportement de ces deux matériaux se trouve en figure 3.14.

La suite de calculs effectuée sur cet exemple contient 200 simulations S_1 suivies de 2 simulations S_2.
Les paramètres de la méthode de réduction de modèles sont :

- pour toutes les simulations S_1 : $\epsilon_R = 10^{-4}$ et $\varepsilon_{POD} = 10^{-5}$.
- pour la première simulation S_2 où la base doit être adaptée, les paramètres sont inchangés.

$\{p_1\}$ Valeur des paramètres matériau utilisés pour la simulation S_1		$\{p_2\}$ Valeur des paramètres matériau utilisés pour la simulation S_2	
$E(MPa)$	2.10^5	$E(MPa)$	2.10^5
ν	0.3	ν	0.3
$R_0(MPa)$	6.10^4	$R_0(MPa)$	100
$H(MPa)$	1000	$H(MPa)$	1000

TAB. 3.1 – Valeur des paramètres matériau du modèle élastoplastique linéaire.

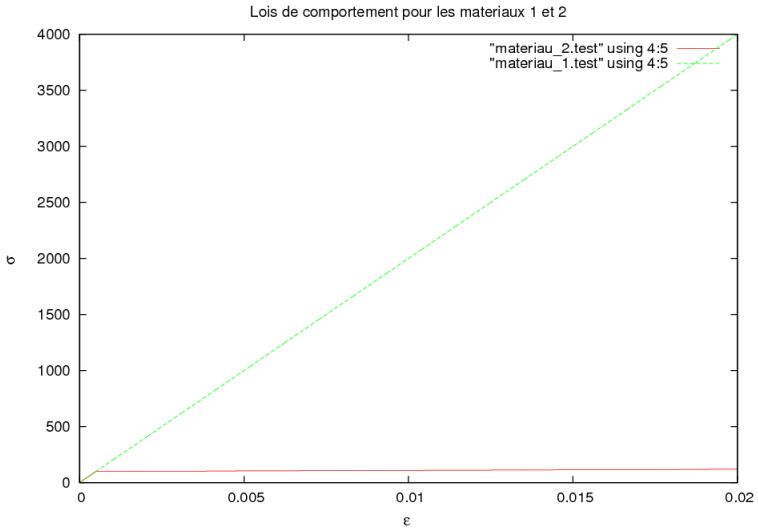


FIG. 3.14 – Courbes de comportements des matériaux 1 (en vert) et 2 (en rouge).

- pour la seconde simulation S_2 où la base doit être figée, d'après la remarque 1 de la section 2.1.2, il faut choisir ϵ_R très grand. Dans notre cas, nous choisissons $\epsilon_R = 10^2$.

3.6.4 Résultats et analyse

Les courbes en figure FIG. 3.15 représentent l'erreur sur la plasticité cumulée en fonction du nombre de simulations S_1 contenues dans la suite de

calculs pour deux valeurs du facteur d'oubli $\gamma = 1$ et $\gamma = 0.5$.

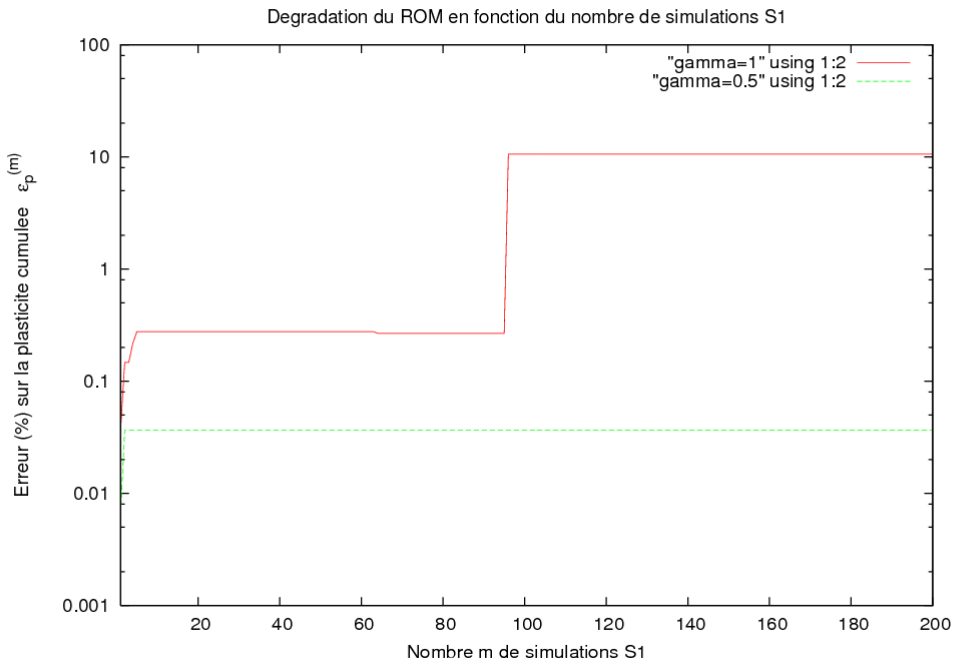


FIG. 3.15 – Dégradation du modèle d'ordre réduit en présence d'évènements récurrents.

Nous constatons qu'en introduisant un facteur d'oubli $\gamma = 0.5$, la dernière adaptation a bien été réalisée car l'erreur de prévision de la dernière simulation est faible (0.04%) quelque soit le nombre de simulations S_1 effectuées. De plus, on montre que le facteur d'oubli a pu permettre d'améliorer considérablement la qualité du calcul puisque l'erreur obtenue sans facteur d'oubli ($\gamma = 1$) était plus forte (10%) à partir de 96 simulations S1 effectuées, suite à l'accumulation de 96 simulations identiques. Il y a eu dégradation du calcul due à la perte d'une fonction de forme. Nous avons ainsi mis en évidence sur un exemple que la précision d'un calcul se détériore avec l'accumulation d'évènements récurrents, et que l'introduction d'un facteur d'oubli permet de résoudre ce problème. A présent, nous allons appliquer la solution proposée à différents problèmes inverses dans la section suivante.

3.7 Résolution de problèmes inverses à évènements récurrents

3.7.1 Problème inverse

Le problème inverse sera traité avec un algorithme classique Simplex [Nelder and Mead, 1965], déjà implémenté dans ZéBuLoN. L'organigramme de cet algorithme est présenté sur la figure suivante (FIG. 3.16).

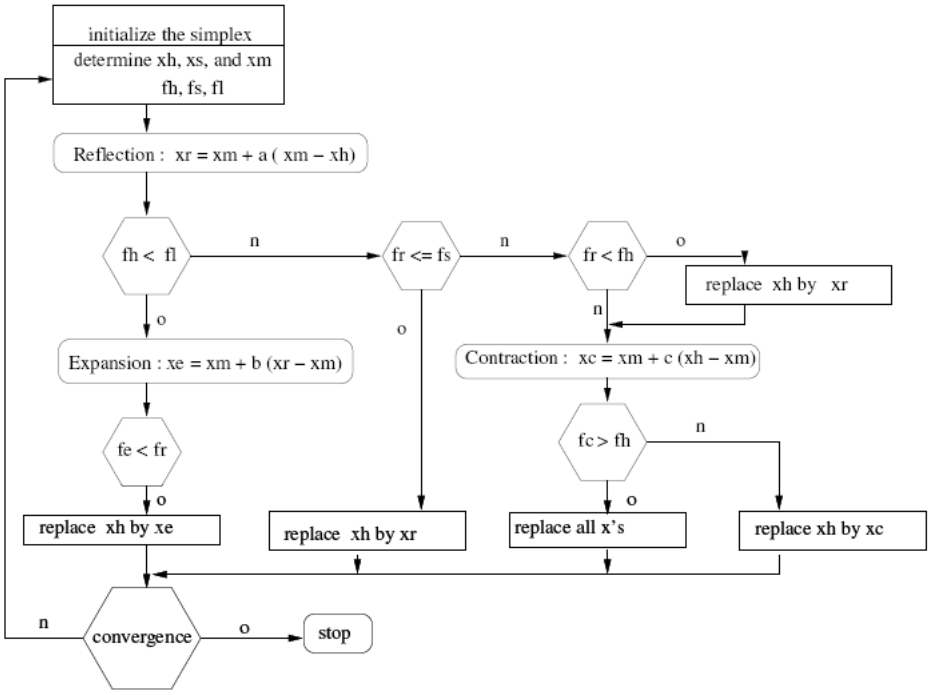


FIG. 3.16 – Organigramme de la méthode Simplex [Nelder and Mead, 1965].

À chaque itération d'optimisation, nous comparons donc le fichier contenant les valeurs des variables d'optimisation Y_{sim} obtenus par la simulation en modèle réduit en cours, avec le fichier contenant les valeurs des déplacements Y_{ref} de référence. La fonction à optimiser (l'erreur) par la méthode Simplex est alors la suivante :

$$F = \frac{1}{N} \sum_{k=1}^{N} \frac{||Y_k - \overline{Y}_k||}{Y_{ref_k}} \tag{3.91}$$

où : Y_k sont les valeurs des variables d'optimisation du fichier le plus petit

(simulation ou référence), $\overline{Y}_k$ sont les valeurs interpolées du fichier le plus petit, Y_{ref_k} sont les valeurs du fichier de référence.

Dans les deux sections suivantes, nous présenterons deux exemples de résolution de problèmes inverses contenant des évènements récurrents, et mettrons en œuvre la solution proposée pour atténuer les évènements récurrents. Rappelons (paragraphe 3.6.2) que l'efficacité du traitement en base réduite est mesurée par le nombre de calculs en base complète nécessaires pour obtenir une prévision de la qualité désirée. Ainsi, pour chaque exemple, l'objectif est de comparer le nombre de calculs en base complète que nous noterons N_g (le nombre de résolutions de problèmes linéaires globaux) pour chaque méthode proposée : la méthode Éléments Finis, la méthode APR, et la méthode APR avec atténuation des évènements récurrents.

3.7.2 Cas d'étude : fil conducteur

Dans ce cas d'étude, dont nous rappelons le maillage et ses conditions aux limites ci-dessous (FIG. 3.17), les variables d'optimisation sont les déplacements U_2. Les coefficients de la relation de comportement sont optimisés afin

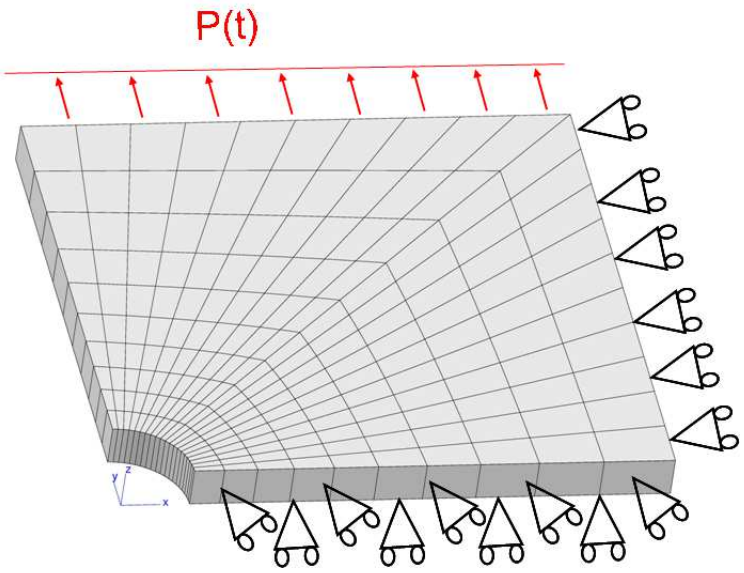


FIG. 3.17 – Maillage d'un huitième de l'éprouvette et conditions aux limites.

de réduire l'écart entre un déplacement calculé et un déplacement donné. Généralement, ces déplacements sont obtenus par des essais. Nous considérons ici un déplacement fourni par une simulation de référence.

Le recalage de modèle consiste donc à résoudre un problème d'optimisation en partant de valeurs de paramètres initiaux, afin de trouver des valeurs de paramètres de référence. Ces valeurs sont fournies dans le tableau TAB. 3.7.2. Nous ne cherchons pas à recaler la valeur du coefficient de poisson ν.

$\{p\}$	Initial	Référence
$E(MPa)$	1.10^5	2.10^5
ν	0.3	0.3
$R_0(MPa)$	300	150
$H(MPa)$	2.10^5	2.10^4

La loi de comportement est représentée sur la figure FIG. 3.18.

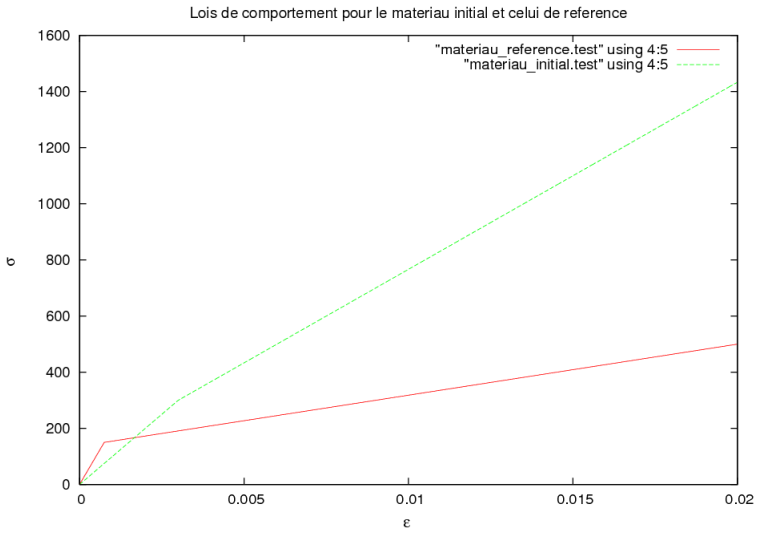


FIG. 3.18 – Courbes de comportement du matériau initial (en vert) et de celui de référence (en rouge).

Sur chacun de ces matériaux, nous imposons une pression $P(t) = 22.08t$.

Ce sont les trois paramètres E, R_0, et H que l'on va chercher à recaler, afin de recaler la solution représentée en figure FIG. 3.19.

Premièrement, un processus d'optimisation en base figée (ϵ_R grand) issue de la première simulation a permis de constater que le processus d'optimisation ne converge pas, et donc que le recalage des paramètres n'arrivait pas à se faire, comme le montrent respectivement les figure FIG. 3.20 et 3.21. Les

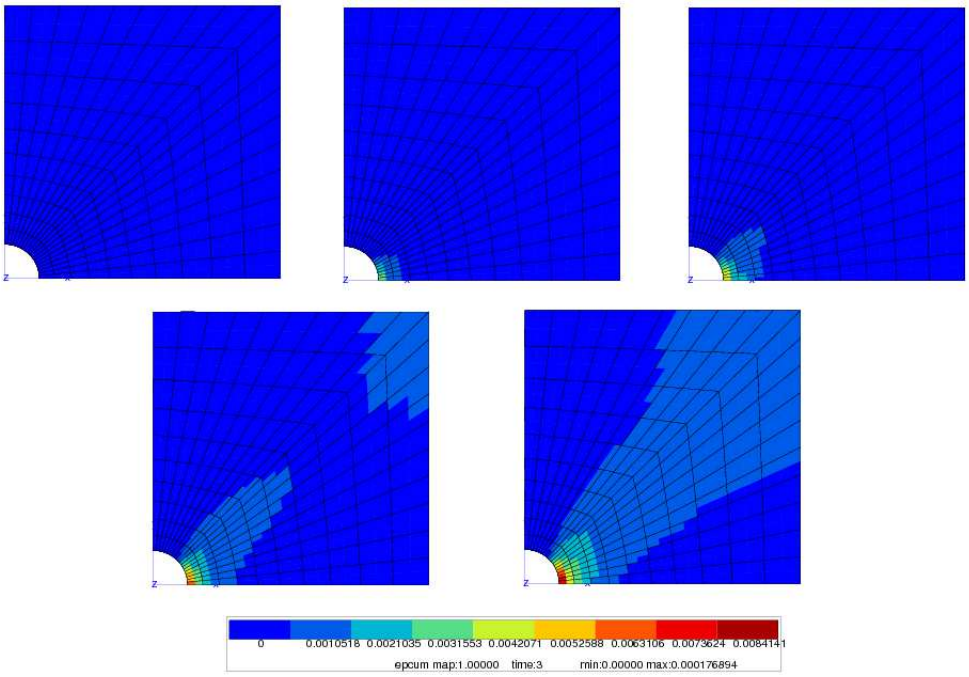


FIG. 3.19 – Solution à recaler : plasticité cumulée au cours de la simulation.

paramètres de la méthode APR sont $\epsilon_R = 10^4$ et $\varepsilon_{POD} = 10^{-8}$.

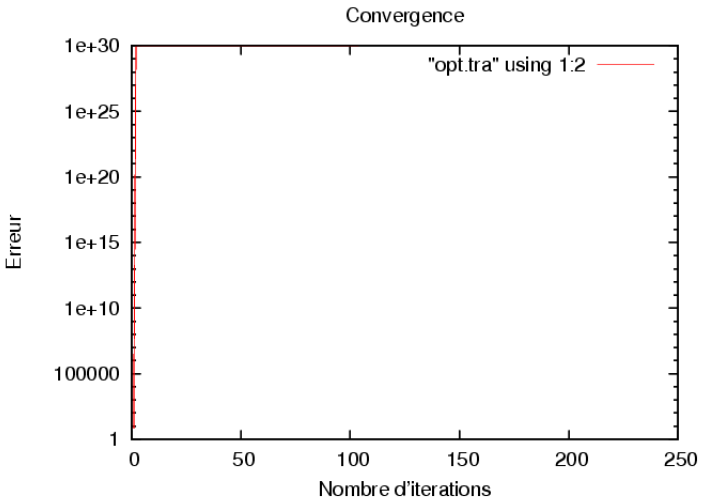


FIG. 3.20 – Non convergence de l'erreur avec une base fiée et $\gamma = 1$.

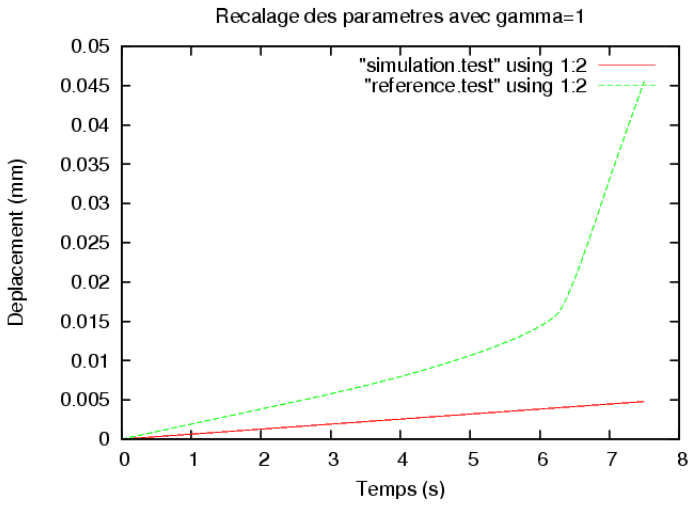


FIG. 3.21 – Échec du recalage des paramètres avec une base figée et $\gamma = 1$.

Ainsi, si le modèle est mauvais, puisque la base n'est pas adaptée, l'optimisation des paramètres échoue.

Par contre, lorsqu'on ne fige pas la base ($\epsilon_R = 5.10^{-3}$) la méthode APR

adapte le modèle d'ordre réduit afin de garantir un niveau d'erreur inférieur à une limite donnée. Nous réalisons un processus d'optimisation en modèle d'ordre réduit avec adaptation de la base, en introduisant un facteur d'oubli $\gamma = 0.9$. Un schéma illustre les résultats obtenus en figure FIG. 3.22. L'optimisation converge après 150 itérations. A la fin de la 1ère simulation, la base a été adaptée 3 fois. A la fin de la dernière simulation, elle a été adaptée 1263 fois pour l'ensemble de la suite de simulations.

Un autre exemple est donné en (FIG. 3.22) avec un facteur d'oubli $\gamma = 0.5$. L'optimisation converge au bout de 146 itérations, et la base a été adaptée 996 fois pour l'ensemble de la suite de simulations.

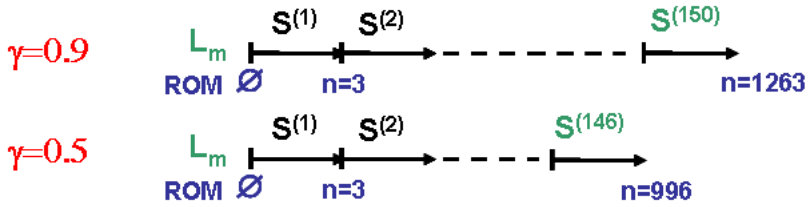


FIG. 3.22 – Nombre de simulations et d'adaptations pour différentes valeurs de γ.

Ainsi, le nombre d'adaptations du modèle d'ordre réduit dépend non seulement de la valeur de ε_{POD}, mais aussi du facteur d'oubli γ.

De plus, si la qualité est suffisante, l'optimisation convergera vers l'optimum quelle que soit la valeur de ε_{POD}, comme l'on peut le voir sur la courbe de convergence de l'erreur (FIG. 3.23) et la courbe de recalage en (FIG. 3.24), ces courbes étant obtenues avec $\gamma = 1$.

Ainsi, pour analyser l'importance des évènements récurrents et le rôle du facteur d'oubli γ, il va falloir comparer le nombre de systèmes globaux linéaires Éléments Finis résolus pour différents γ. Celui ci sera d'autant plus élevé que la présence d'évènements récurrents sera importante. Nous comparons donc un calcul d'optimisation Éléments Finis avec un calcul d'optimisation en base réduite avec différents γ. Le résultat de ces différents processus est donné dans le tableau 3.2.

Nous pouvons remarquer que l'on obtient un minimum de problèmes globaux traités en choisissant un facteur d'oubli $\gamma = 0.5$ (ce nombre est divisé par $10,35$ par rapport au problème Éléments Finis, et par $1,74$ par rapport au problème réduit avec $\gamma = 1$).

Bien évidemment, il est probablement possible de trouver un modèle d'ordre

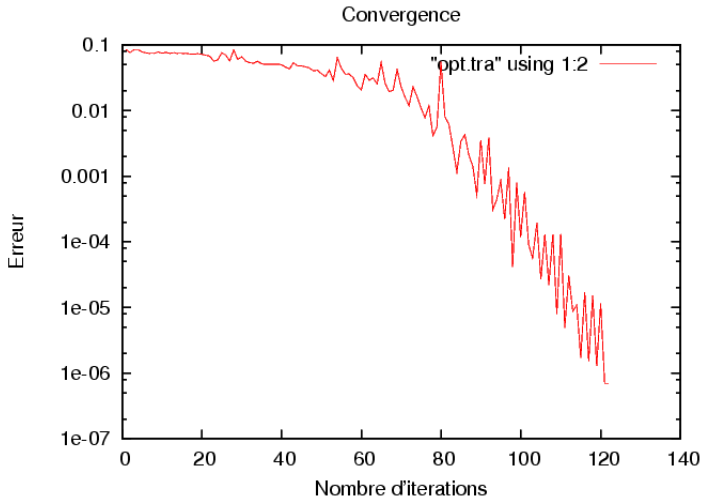


FIG. 3.23 – Convergence de l'erreur avec une base adaptée et $\gamma = 1$.

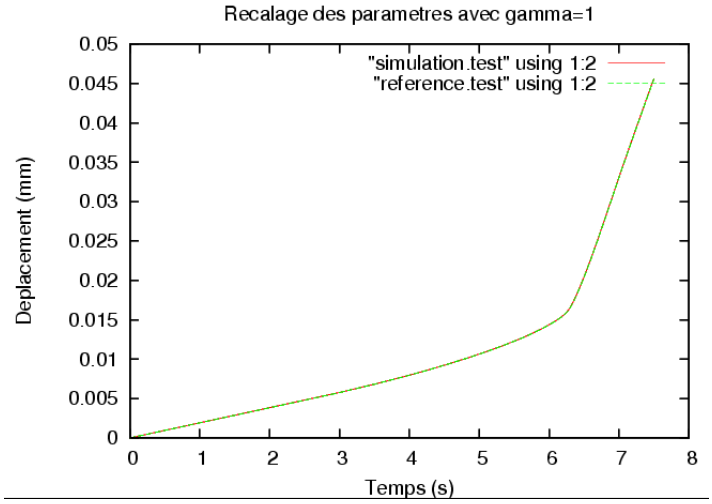


FIG. 3.24 – Recalage des paramètres réussi avec une base adaptée et $\gamma = 1$.

Méthode utilisée	Éléments Finis	APR				
γ	-	1	0.9	0.5	0.1	0
Nombre d'itérations du processus d'optimisation	130	122	150	146	156	160
Nombre de résolutions de problèmes linéaires globaux	23432	3946	3098	2263	2396	3211
Gain en nombre de résolutions de problèmes linéaires globaux par rapport à la méthode EF	-	5.94	7.56	10.35	9.78	7.3
Gain en nombre de résolutions de problèmes linéaires globaux par rapport à la méthode APR avec $\gamma = 1$	-	-	1.27	1.74	1.65	1.23

TAB. 3.2 – Comparaison du nombre de problèmes linéaires globaux résolus pour différentes méthodes utilisées dans le processus d'optimisation.

réduit figé permettant de réaliser la suite de simulations requise pour le problème inverse. La stratégie que nous proposons ne consiste pas à supposer l'existence d'un tel modèle. Au cours du processus d'optimisation, nous recherchons les paramètres optimaux et nous construisons un modèle d'ordre réduit évolutif.

Nous proposons à présent de traiter un problème avec localisation des déformations, afin de montrer que la méthode proposée est indépendante du choix de cette loi de comportement.

3.7.3 Modèle d'endommagement

3.7.4 Présentation du modèle

Voici un autre cas d'étude sur lequel nous avons choisi de travailler, afin d'avoir un modèle d'ordre réduit très dépendant des paramètres à optimiser. Cette étude a fait l'objet d'une publication [Ryckelynck et al., 2011], que le lecteur pourra trouver en annexe 4.5.Un modèle d'endommagement proposé par Rousselier [Rousselier, 1987] induit une localisation des déformations et une réponse de l'éprouvette qui sont totalement influencées par les paramètres de la loi de comportement. Pour éviter toute discussion sur la sensibilité de la localisation vis à vis du maillage, nous n'avons considéré qu'un

seul maillage de l'éprouvette. Pour plus de détails sur les problèmes de sensibilité du maillage, consulter [Pijaudier-Cabot and Benallal, 1993].
L'éprouvette utilisée pour ce modèle est représentée sur la figure FIG. 3.25. Sur cette même figure, on peut trouver les conditions aux limites que l'on a imposées : sur toute la partie inférieure de l'éprouvette, on impose un déplacement $U2 = 0$. On impose également $U1 = 0$ sur le coin inférieur droit. Sur toute la partie supérieure de l'éprouvette, on impose un déplacement $\bar{u}_0(t) = 25t$.

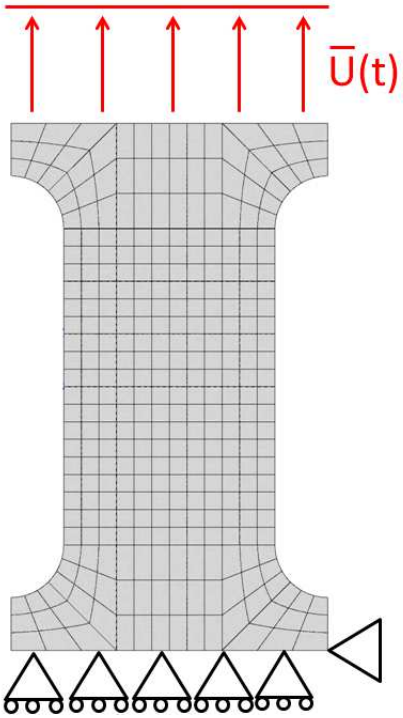


FIG. 3.25 – Maillage de l'éprouvette utilisée.

Le matériau est un acier de cuve d'un réacteur nucléaire.
Dans ce cas d'étude, les variables d'optimisation sont les efforts. De la même manière que pour le premier modèle, nous considérons ici un effort donné par une simulation de référence.
On considère un problème élastoplastique avec prévision des dommages par une loi de Rousselier. La relation de comportement est élastoplastique avec écrouissage isotrope. Le modèle de Rousselier décrit l'endommagement dû à la croissance plastique de cavités dans un métal, et permet de modéliser la fissuration et la rupture ductile. Celle-ci impliquant des niveaux de déformation assez important, nous nous plaçons dans le cadre de grandes transformations.

3.7.5 Équations du modèle

Le modèle utilisé pour ce travail est basé sur la définition d'un référentiel corotationnel. Cela nous permet de pouvoir conserver un formalisme des petites déformations pour écrire les équations de la loi de comportement.

Le gradient de vitesse $\underset{\sim}{\boldsymbol{L}} = \dot{\underset{\sim}{\boldsymbol{F}}}.\underset{\sim}{\boldsymbol{F}}^{-1}$ est découpé en une partie symétrique ($\underset{\sim}{\boldsymbol{D}}$) et une partie anti-symétrique ($\underset{\sim}{\boldsymbol{\Omega}}$). La rotation définissant le référentiel corotationnel est donné par l'équation suivante :

$$\dot{\underset{\sim}{\boldsymbol{Q}}}_c = \underset{\sim}{\boldsymbol{\Omega}}.\underset{\sim}{\boldsymbol{Q}}_c \quad \text{avec} \quad \underset{\sim}{\boldsymbol{Q}}_c(t=0) = \underset{\sim}{\mathbf{1}} \tag{3.92}$$

Le tenseur des contraintes tournées ($\underset{\sim}{\boldsymbol{\sigma}}_c = \underset{\sim}{\boldsymbol{Q}}_c.\underset{\sim}{\boldsymbol{\sigma}}.\underset{\sim}{\boldsymbol{Q}}_c^T$) et le tenseur taux de déformation ($\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_c = \underset{\sim}{\boldsymbol{Q}}_c.\underset{\sim}{\boldsymbol{D}}.\underset{\sim}{\boldsymbol{Q}}_c^T$) sont utilisés pour formuler la loi de comportement. La dérivée du tenseur des contraintes de Cauchy est la dérivée de Jaumann. $\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_c$ se décompose de la manière suivante :

$$\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_c = \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_p + \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_e \tag{3.93}$$

où $\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_p$ est le tenseur taux de déformation plastique et $\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_e$ le tenseur taux de déformation élastique donné par $\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_e = \underset{\sim}{\boldsymbol{E}} : \dot{\underset{\sim}{\boldsymbol{\sigma}}}_c$ (hypoélasticité). $\underset{\sim}{\boldsymbol{E}}$ est le tenseur de Hooke. Les variables internes sont : ε_e et on note p la plasticité cumulée.

Le critère de limite élastique est donné par :

$$\Phi = \frac{\overline{\sigma}_c}{1-f} + \sigma_1 f D \exp\left(\frac{\mathrm{tr}(\underset{\sim}{\boldsymbol{\sigma}}_c)}{3(1-f)\sigma_1}\right) - \sigma_f(p) \tag{3.94}$$

où $\overline{\sigma}_c$ est l'invariant de von Mises $\underset{\sim}{\boldsymbol{\sigma}}_c$, $\mathrm{tr}(\underset{\sim}{\boldsymbol{\sigma}}_c)$ est la trace du tenseur des contraintes tournées et $\sigma_f(p)$ la contrainte d'écoulement du matériau non endommagé ; il s'agit d'une loi d'écrouissage isotrope qui dépend de la déformation plastique cumulée p de la manière suivante :

$$\sigma_f(p) = R_0 + Q(1 - e^{-bp}) \tag{3.95}$$

où R_O est la limite élastique, et Q et b sont 2 coefficients représentant l'évolution des variables d'écrouissage isotrope. f est la variable d'endommagement correspondant à la fraction volumique de porosité. σ_1 et D sont les paramètres du modèle (D étant le paramètre pilotant l'endommagement). En considérant la loi de normalité, le tenseur taux de déformation plastique s'exprime de la manière suivante :

$$\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_p = (1-f)\dot{p}\frac{\partial \Phi}{\partial \underset{\sim}{\boldsymbol{\sigma}}_c} \tag{3.96}$$

Le multiplicateur plastique (ou taux de plasticité cumulée) $\dot{p}$ s'écrit $\dot{p} = \sqrt{\frac{2}{3}\dot{\underset{\sim}{\varepsilon}}'_p : \dot{\underset{\sim}{\varepsilon}}'_p}$ où $\dot{\underset{\sim}{\varepsilon}}'_p$ est le déviateur de $\dot{\underset{\sim}{\varepsilon}}_p$. L'évolution de l'endommagement est donné en utilisant la conservation de la masse, ce qui donne :

$$\dot{f} = (1-f)\mathrm{tr}(\dot{\underset{\sim}{\varepsilon}}_p) \tag{3.97}$$

La nucléation est modélisée simplement par une porosité initiale donnée f_0. Il n'y a pas de nucléation continue dans ce modèle. La coalescence des cavités est à peu près modélisée par la localisation des déformations. En dépit de ces hypothèses, le modèle de Rousselier est capable de décrire l'endommagement de structures complexes [Besson et al., 2001]. Une reformulation non-locale du modèle de Rousselier a été proposée en [Lorentz et al., 2008] pour améliorer la robustesse de l'intégration numérique et éviter la dépendance des résultats au maillage, causé par l'adoucissement par déformation.
Le modèle que nous utilisons introduit une dépendance des résultats au maillage, mais nous avons choisi de travailler en maillage figé.

La formulation 1.6, en transformation finies, devient :

$$\begin{aligned}\int_{\Omega^0\times\mathcal{P}} \underset{\approx}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*,\underline{\boldsymbol{u}}) \,:\, \underset{\approx}{\boldsymbol{\Sigma}}(\underset{\approx}{\boldsymbol{F}}(\underline{\boldsymbol{u}}),\, \tau\le t;\{p\})\;\; d\Omega^0 dp \\ - \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^* \,.\, \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}},t,\{p\})\;\; d\Gamma^0 dp \\ = \int_{\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_v d\Omega^0 dp + \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_b d\Gamma^0 dp \\ \forall \underline{\boldsymbol{u}}^* \in \mathcal{V}_m \quad \forall\, \underline{\boldsymbol{u}}^* \in \mathcal{V},\;\; \forall t\in\,]0,T]\end{aligned} \tag{3.98}$$

3.7.5.1 Paramètres du modèle à recaler

Les paramètres à recaler sont fournis dans le tableau TAB. 3.3.

$\{p\}$	Initial	Référence
R_0	300	486.5
Q	500	616.3
b	10	5.73
D	1	2.66

TAB. 3.3 – Paramètres du modèle à recaler.

Q et b sont 2 coefficients représentant l'évolution des variables d'écrouissage isotrope, R_0 est la limite d'élasticité et D le paramètre pilotant l'endommagement. Celui ci étant assez faible, il s'agit d'un matériau ductile. Ce sont

ces 4 paramètres que l'on va chercher à recaler, afin de recaler la solution représentée en figure FIG. 3.26.

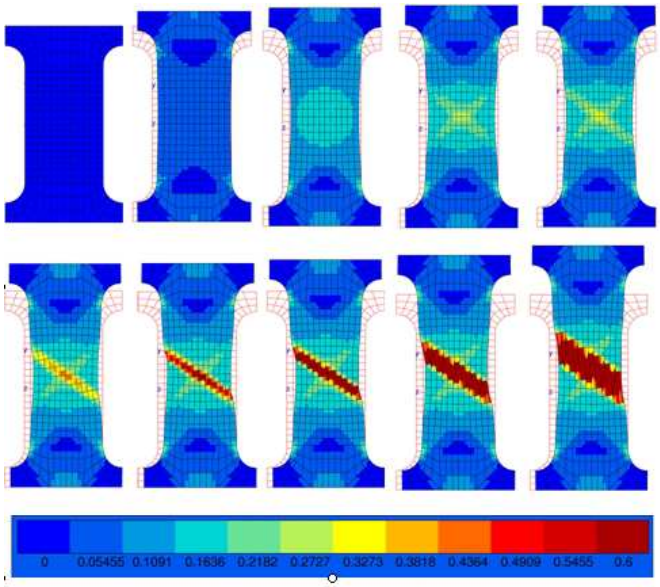


FIG. 3.26 – Solution à recaler : plasticité cumulée au cours du temps.

Les paramètres de la méthode APR sont $\epsilon_R = 5.10^{-3}$ et $\varepsilon_{POD} = 10^{-4}$.

3.7.5.2 Résultats

Le tableau de comparaison du nombre de problèmes linéaires globaux résolus pour différents processus d'optimisation est donné ci-dessous (TAB. 3.4).

Comme pour le premier cas que nous avons étudié, la valeur optimale de γ est 0.5. Et la qualité du processus d'optimisation n'est pas altérée, puisque non seulement le recalage des paramètres se fait parfaitement, comme le prouve la figure FIG. 3.27, mais le nombre d'itérations pour atteindre les paramètres de référence avec $\gamma = 0.5$ est plus faible (85 itérations) par rapport à l'optimisation effectuée sans facteur d'oubli ($\gamma = 1$), pour laquelle il faut 110 itérations avant d'atteindre la convergence du processus d'optimisation.

3.7.6 Conclusion

La méthode proposée a été mise en oeuvre sur des problèmes compliqués et variés : elle a été appliquée à différents types de comportements mécaniques :

Méthode utilisée	Éléments Finis	APR			
γ	-	1	0.9	0.5	0
Nombre d'itérations du processus d'optimisation	104	110	106	85	93
Nombre de résolutions de problèmes linéaires globaux	108762	40155	39302	38771	53387
Gain en nombre de résolutions de problèmes linéaires globaux par rapport à la méthode EF	-	2.71	2.77	2.8	2.04

TAB. 3.4 – Comparaison du nombre de problèmes linéaires globaux résolus pour différents processus d'optimisation.

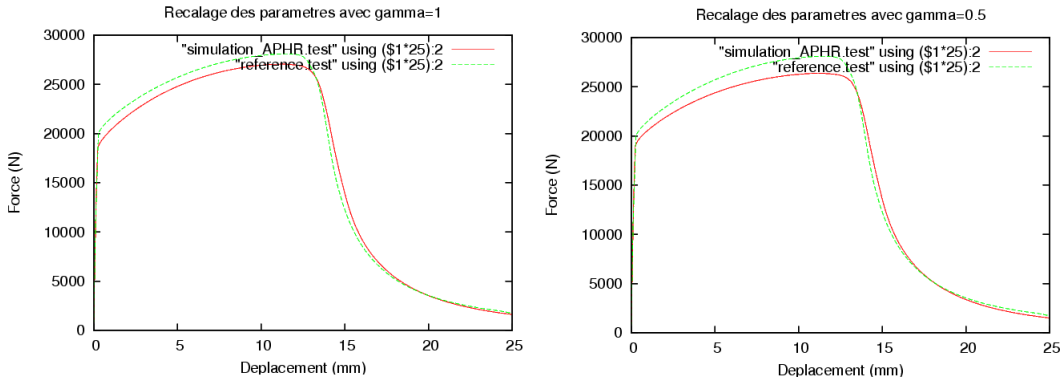


FIG. 3.27 – Recalage des paramètres pour $\gamma = 0.5$ et $\gamma = 1$.

élastoplastique ou plasticité endommageable. La méthode adaptative permet de traiter de façon séquentielle des problèmes paramétrés fortement non-linéaires du type résolution de problèmes inverses. Nous avons ainsi traité des problèmes d'optimisation contenant de nombreuses itérations (> 100), et avons remarqué que dans ce cas, des évènements récurrents pouvaient apparaître suite à l'adaptation de la base réduite. Nous avons donc apporté une solution efficace à l'atténuation de ces évènements récurrents en développant un algorithme de réduction adaptative dédié aux problèmes d'optimisation avec prise en compte de la présence d'évènements récurrents nuisibles à l'efficacité du processus.
Le prochain chapitre sera consacré à l'utilisation des méthodes APR et APHR

non plus pour des suites de simulations, mais pour la résolution de simulations simultanées.

Chapitre 4

Simulations multidimensionnelles

Jusqu'à présent, nous nous sommes intéressés à des processus purement incrémentaux séquentiels : il s'agissait d'effectuer une suite de simulations sur un axe de temps virtuel qui différaient les unes des autres par une variation de paramètres. Cela pouvait également être vu comme un parcours de l'espace des paramètres sur un axe de temps virtuel (FIG. 4.1).

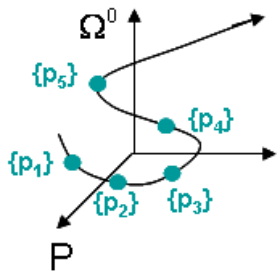


FIG. 4.1 – Parcours de l'espace des paramètres $\mathcal{P}$ sur un axe de temps virtuel.

Nous avons vu dans les équations (3.2) et (3.3) que la réduction de modèle permettait de séparer les variables d'espace et de temps, et ainsi, d'écrire $\underline{\boldsymbol{u}}$ en représentation à variables séparées sur l'espace (Ω^0, t') de la manière suivante :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,, t; \{p\}_m) = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_k^{(n)}(\underline{\boldsymbol{X}}\,)\, a_k^{(n,m)}(t; \{p\}_m) \tag{4.1}$$

Ou encore, grâce à l'introduction d'un temps global :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t') = \sum_{k=1}^{k=s} \underline{\boldsymbol{\psi}}_{k}^{(n)}(\underline{\boldsymbol{X}}\,)\, a_k^{(n)}(t') \tag{4.2}$$

Mais l'état mécanique d'un système peut être vu comme une fonction dépendant non seulement du temps et des variables de l'espace, mais aussi des paramètres tels que les paramètres matériau. Ainsi, en considérant que le paramètre est aussi une coordonnée, nous sommes dans le cadre d'une simulation multidimensionnelle, et $\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t;\{p\}_m)$ s'écrit maintenant $\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t,\{p\})$.

Nous considérons que $\underline{\boldsymbol{X}}$ est devenu la nouvelle coordonnée $\underline{\tilde{\boldsymbol{X}}} = (\underline{\boldsymbol{X}}\,,\{p\})$, un point dans un domaine multidimensionnel espace/paramètres $\tilde{\Omega} = (\Omega^0 \times \mathcal{P})$. Il s'agit en fait de dupliquer le maillage du problème, autant de fois que l'on a de cas à traiter. Soit $\tilde{\Omega}$ l'espace du maillage dupliqué (FIG. 4.2).

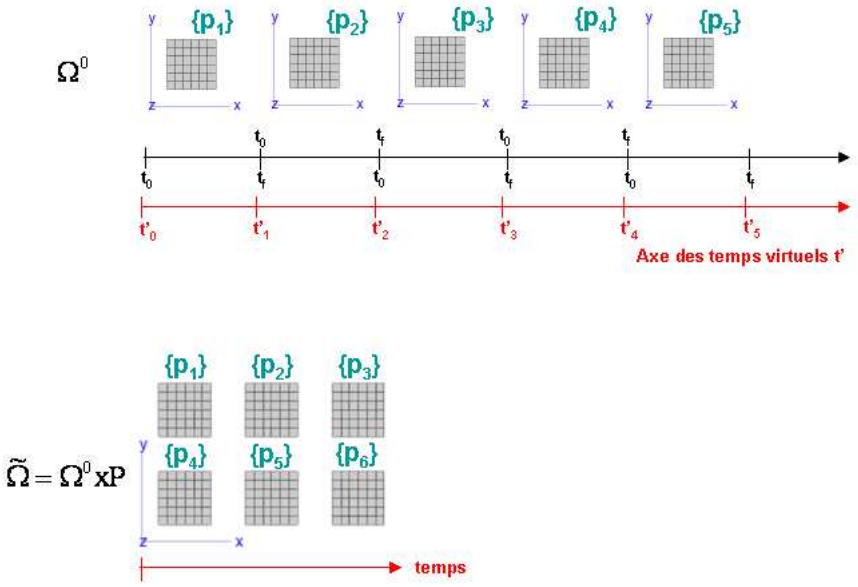


FIG. 4.2 – Suites de maillages sur l'espace Ω^0 et maillage dupliqué sur l'espace $\tilde{\Omega} = \Omega^0 \times \mathcal{P}$.

Considérant cette écriture, le mode $\underline{\boldsymbol{\psi}}_k(\underline{\boldsymbol{X}}\,)$ de la méthode APHR devient donc $\underline{\boldsymbol{\psi}}_k(\underline{\tilde{\boldsymbol{X}}}\,)$ et est donc dépendant de $\underline{\boldsymbol{X}}$ et de $\{p\}$. C'est donc un mode de l'espace $\tilde{\Omega}$. La réduction de modèle permet de séparer les variables d'espaces et les variables du temps (2.1.2). Elle doit maintenant être appliquée sur ce nouvel espace $\tilde{\Omega} = (\Omega^0 \times \mathcal{P})$ afin de séparer les variables d'espace et les variables de paramètres. Le but du travail est de proposer un algorithme dans

lequel la séparation de variables entre $\underline{\boldsymbol{X}}$ et $\{p\}$ sera effectuée. On cherchera donc à écrire ce mode sous la forme :

$$\underline{\boldsymbol{\psi}}_k(\underline{\tilde{\boldsymbol{X}}}) = \sum_j^{N_l} \underline{\boldsymbol{\Phi}}_{kj}(\underline{\boldsymbol{X}}) g_{jk}(\{p\}) \tag{4.3}$$

Il s'agit en fait de passer d'un système multidimensionnel à des simulations simultanées. On cherche à écrire $\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, t, \{p\})$ sur un système particulier : une somme de systèmes découplés qui correspond à une simulation simultanée de différents problèmes. Pour un incrément de temps donné, plusieurs valeurs de paramètres seront traitées de manière simultanée sur l'ensemble du système. Cela peut paraître aberrant de traiter de façon couplée des problèmes découplés. Mais on aboutit à une méthode plus efficace en traitant le problème multidimensionnel dans sa globalité, plutôt que de façon séquentielle (comme la représentation par une suite de simulations sur un axe de temps virtuel que l'on présentait dans le chapitre **??**). Ainsi, bien qu'il semble plus compliqué d'envisager l'ensemble des problèmes en même temps, paradoxalement, on obtient une meilleur efficacité numérique, comme le prouveront les résultats des sections 4.4.7 et 4.5. Nous proposons donc un solveur linéaire pour les systèmes matriciels de type bloc sur la diagonale. En effet, les problèmes étant découplés, chaque bloc de la matrice de rigidité correspond à un problème. Nous obtiendrons ainsi une résolution simultanée de problèmes découplés.

Dans ce chapitre, la première section sera consacrée à la reformulation générale d'un problème mécanique dans le cadre de simulations multidimensionnelles. L'algorithme dans lequel s'effectuera la séparation variables d'espace/variables de paramètres sera détaillé dans une seconde partie. Puis en troisième et en quatrième partie, nous présenterons deux exemples d'utilisation de cet algorithme. Ces deux exemples nous permettront de conclure quant à l'efficacité de ce nouvel algorithme pour réaliser des simulations simultanées par rapport à l'utilisation d'une méthode classique de réduction de modèle dans le cadre de suites de simulations

4.1 Reformulation du problème mécanique dans le cadre de simulations multidimensionnelles

4.1.1 Formulation du modèle continu

Le but de la méthode proposée est d'estimer l'état mécanique comme une fonction définie dans un espace multidimensionnel $\Omega^0 \times \mathcal{P} \times]0,T]$, où, rappelons le :

- Ω^0 est la configuration de référence du système mécanique,
- $\mathcal{P}$ l'espace multiple relatif aux paramètres du modèle,
- $]0,T]$ l'intervalle de temps.

Nous nous plaçons dans le cadre de grandes déformations.

Rappelons également que $\partial\Omega^0$ est la frontière du domaine Ω^0 qui se décompose de manière classique $\partial_U\Omega^0 \cup \partial_f\Omega^0$. Sur la partie $\partial_f\Omega^0$, on impose un champ d'efforts donné $\underline{\boldsymbol{f}}\,(.,t;.)$ dépendant du temps t.

Soit $\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t,\{p\})$ le champ de déplacements au temps t défini sur le domaine $\Omega^0 \times \mathcal{P}$.
Ce champ est représenté en utilisant la méthode des variables séparées de la manière suivante :

$$\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t,\{p\}) = \sum_{k=1}^{s}\left(\sum_{j=1}^{j=N_l} \underline{\boldsymbol{\Phi}}_{kj}(\underline{\boldsymbol{X}}\,)g_{kj}(\{p\})\right)a_k(t) + \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}\,,t) \qquad (4.4)$$

$$\forall \underline{\boldsymbol{X}} \in \Omega^0,\ \forall\{p\} \in \mathcal{P},\ \forall t \in]0,T]$$

où :

- $\underline{\boldsymbol{\Phi}}_j(\underline{\boldsymbol{X}}\,)$ est un champ de déplacements défini sur $H^1(\Omega^0)$,
- $g_j(\{p\})$ une fonction scalaire définie sur $L^2(\mathcal{P})$,
- $a_j(t)$ une fonction scalaire continue du temps,
- $\underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}\,,t)$ est une fonction donnée qui vaut zéro presque partout et telle que les conditions de Dirichlet définies sur $\partial_U\Omega^0 \times \mathcal{P}$ soient satisfaites.

Sur la partie $\partial_u\Omega^0$, on impose un déplacement $\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,.,t) = \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}\,,t)\ \forall\underline{\boldsymbol{X}} \in \partial_U\Omega^0$. Ainsi, $\underline{\boldsymbol{\Phi}}_j(\underline{\boldsymbol{X}}\,) = 0$ sur $\partial_U\Omega^0$.
Le champ de déplacements appartient à l'espace $\mathcal{U}_m$ des fonctions affines,

défini par :

$$\begin{aligned}
&\mathcal{U}_m = \{\underline{\boldsymbol{u}}\,(.,t,.) \in H^1(\Omega^0) \bigotimes L^2(\mathcal{P}) | \\
&\exists(\underline{\boldsymbol{\Phi}}_j)_{j=1,\ldots,\gamma}\ \ \exists(g_j)_{j=1,\ldots,\gamma},\ \ \exists(a_j)_{j=1,\ldots,\gamma} \\
&\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t,.) = \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}\,,t)\ \ \forall \underline{\boldsymbol{X}}\ \in \partial_U\Omega^0 \qquad (4.5)\\
&\underline{\boldsymbol{u}}\,(\underline{\boldsymbol{X}}\,,t,\{p\}) = \sum_{k=1}^{s}\left(\sum_{j=1}^{j=N_l} \underline{\boldsymbol{\Phi}}_{kj}(\underline{\boldsymbol{X}}\,)g_{kj}(\{p\})\right)a_k(t) + \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}\,,t)\}
\end{aligned}$$

On peut introduire un espace de fonctions, $\mathcal{V}_m$, espace des champs admissibles à zéro, défini par :

$$\mathcal{V}_m = \{\underline{\boldsymbol{u}}^*(.,t,.) \in H^1(\Omega^0) \bigotimes L^2(\mathcal{P}) \mid \underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}}\,,t,.) = 0\ \ \forall \underline{\boldsymbol{X}}\ \in \partial_U\Omega^0\} \quad (4.6)$$

En transformations finies, le problème mécanique s'énonce alors comme suit :

Trouver une estimation du champ de déplacements $\underline{\boldsymbol{u}}\ \in \mathcal{U}_m$ satisfaisant les équations de comportement du milieu continu et le principe des travaux virtuels :

$$\begin{aligned}
\int_{\Omega^0\times\mathcal{P}} \varepsilon(\underline{\boldsymbol{u}}^*, \underline{\boldsymbol{u}}\,)\ :\ \boldsymbol{\Sigma}(\underset{\sim}{\boldsymbol{F}}(\underline{\boldsymbol{u}}\,)\,,\ \tau \le t, \{p\})\ \ d\Omega^0 dp\ \ - \\
\int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^*\ \cdot\ \underline{\boldsymbol{f}}\,(\underline{\boldsymbol{X}}\,,t,\{p\})\ \ d\Gamma^0 dp = \\
\int_{\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{r}}_v d\Omega^0 dp + \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{r}}_b d\Gamma^0 dp \\
\forall \underline{\boldsymbol{u}}^* \in \mathcal{V}_m \quad \forall\, \underline{\boldsymbol{u}}^* \in \mathcal{V},\ \ \forall t \in\,]0,T] \qquad (4.7)
\end{aligned}$$

avec la condition suivante :

$$\int_{\Omega^0\times\mathcal{P}} \underline{\boldsymbol{r}}_v.\underline{\boldsymbol{r}}_v d\Omega^0 dp + \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{r}}_b.\underline{\boldsymbol{r}}_b d\Gamma^0 dp \le \epsilon_R + \varepsilon_h\ \ \forall t \in\,]0,T] \qquad (4.8)$$

4.1.2 Formulation du modèle APHR.

L'utilisation de fonctions tests bien adaptées est le point clef d'une représentation efficace à l'aide de la méthode de séparation de variables.

En principe, l'équation (4.7) n'est pas une équation à variables séparées, car on ne peut pas choisir un tenseur de contrainte $\underset{\sim}{\mathbf{S}}$ à l'aide de variables séparées, sans introduire une nouvelle erreur d'approximation. De plus, à cause de la non-linéarité du problème, il est difficile, voire impossible dans de nombreux cas, de montrer que l'opérateur $\underset{\sim}{\sum}$ est séparable en temps, en espace et en paramètres.

Ainsi, ce n'est pas possible de réécrire les intégrales définies sur $\Omega^0 \times \mathcal{P}$ ou $\partial\Omega^0 \times \mathcal{P}$ comme une somme de produits d'intégrales mono-dimensionnelles définies sur $\mathcal{P}$ et Ω^0 ou $\partial\Omega^0$, comme proposé en [Ammar et al., 2006b] et [Gonzalez et al., 2010].
C'est la raison pour laquelle nous proposons de compléter la méthode d'Hyper Réduction proposée en [Ryckelynck, 2005] et [Ryckelynck, 2009] pour les méthodes de séparation de variables.
Nous appelerons "schéma d'intégration tronquée", ce schéma d'intégration simplifié. La particularité de l'approche proposée est l'extension du schéma d'intégration tronquée aux problèmes admettant des conditions aux limites.
La solution (4.4) sera calculée de manière adaptative.
Contrairement à ce qui est proposé en [Ammar et al., 2006b] et [Ammar et al., 2006a], où la solution et chaque correction sont cherchées sur tout l'intervalle de temps, nous choisissons de travailler avec un schéma incrémental. En effet, nous proposerons en section 4.2 un algorithme à implémenter dans le code de calcul ZéBuLoN, et il s'avère plus facile de modifier un code de calcul standard existant en conservant une approche incrémentale.
L'algorithme adaptatif utilisé pour construire cette solution est un algorithme de prévision-correction. Cet algorithme est donc incrémental : l'intervalle de temps est divisé en plusieurs incréments de temps. On peut prédire l'état mécanique du système à la fin de chaque incrément à condition de connaître l'état au début de l'incrément. A chaque incrément de temps, on adapte le sous-espace des fonctions $g_j(\{p\})$ et $a_j(t)$ si cela s'avère nécessaire.
L'algorithme proposé est similaire à celui proposé en [Gonzalez et al., 2010]. En effet, il est appliqué pas à pas sur l'intervalle de temps en utilisant un schéma d'intégration tronquée.
Supposons que l'on connaît une représentation à variables séparées du système au temps t.
Pendant l'étape de prévision, la représentation à variables séparées des champs définis sur $\Omega^0 \times \mathcal{P}$ est similaire à un modèle d'ordre réduit, dont les variables d'état réduites sont les valeurs des fonctions $(a_j(t))_{j=1,\dots,N_l}$ à la fin de chaque incrément de temps.
Pour pouvoir calculer les variables d'état réduites (si le nombre de celles-ci est assez faible), on peut considérer une restriction des équations d'équilibres

au domaine d'intégration tronquée $\mathcal{W}_\Pi$, sous-domaine de $\Omega^0 \times \mathcal{P}$ (en effet, comme on a peu de variables d'états réduites a, on n'a pas besoin de toutes les équations, donc on effectue une intégration tronquée). Ce domaine d'intégration tronquée est le support des fonctions tests tronquées, relatives à la représentation à variables séparées connue. Il s'agit en fait du RID (section 2.4).

Dans la formulation des équations d'équilibre proposée ci-dessus, les fonctions tests appartiennent à l'espace entier Ω^0.
Mais de nombreuses conditions d'équilibres peuvent être introduites utilisant différents espaces de vecteurs, à condition que le rang de ces équations soit égal au nombre d'inconnues N_l, qui est aussi la dimension du modèle d'ordre réduit défini par la représentation à variables séparées.
Ainsi, de la même manière qu'en [Ryckelynck, 2005] et [Ryckelynck, 2009], nous proposons d'introduire un espace de vecteurs des fonctions tests tronquées, noté $\mathcal{W}_\Pi$ et tel que :

$$
\begin{aligned}
&\mathcal{W}_\Pi = \{\underline{\boldsymbol{u}}^*(.,.) \in H^1(\Omega^0) \bigotimes L^2(\mathcal{P})| \\
&\underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}}, t, .) = 0 \ \forall \underline{\boldsymbol{X}} \in \partial_U \Omega^0 \\
&\underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}}, t, \{p\}) = h(\underline{\boldsymbol{X}}, \{p\}) \sum_{j=1}^{j=\gamma} \underline{\boldsymbol{\Phi}}_j(\underline{\boldsymbol{X}}) g_j(\{p\}) a_j^* \\
&h(\underline{\boldsymbol{X}}, \{p\}) \in C^\infty(\Omega^0 \times \mathcal{P}) \\
&h(\underline{\boldsymbol{X}}, \{p\}) = 0 \ \forall (\underline{\mathbf{X}}, \{p\}) \notin \mathcal{W}_\Pi
\end{aligned}
\tag{4.9}
$$

$$
\min_{h \in \mathcal{C}^\infty(\Omega^0 \times \mathcal{P})} \int_{\mathcal{W}_\Pi} (1-h)^2 d\Omega\}
$$

En utilisant cet espace pour les fonctions tests, les intégrales définies sur l'espace complémentaire de $\mathcal{W}_\Pi$ sont nulles.
Ainsi, le problème de prévision est :

Connaissant $(\underline{\boldsymbol{\Phi}}_j)_{j=1,\ldots,\gamma}$ et $(g_j)_{j=1,\ldots,\gamma}$, trouver $(a_j(t))_{j=1,\ldots,\gamma}$ tels que $\underline{\boldsymbol{u}} \in \mathcal{U}$ minimise η_Π, et tel que :

$$
\eta_\Pi = \int_{\mathcal{W}_\Pi} \underline{\boldsymbol{r}}_v . \underline{\boldsymbol{r}}_v d\Omega^0 dp + \int_{\partial_f \Omega^0 \times \mathcal{P} \cap \mathcal{W}_\Pi} \underline{\boldsymbol{r}}_b . \underline{\boldsymbol{r}}_b d\Gamma^0 dp \tag{4.10}
$$

et

$$\int_{\mathcal{W}_\Pi} \varepsilon(\underline{\boldsymbol{u}}^*, \underline{\boldsymbol{u}}) \ : \ \boldsymbol{\Sigma}(\underset{\sim}{\boldsymbol{F}}(\underline{\boldsymbol{u}}), \tau \leq t; \{p\}) \ d\Omega^0 dp \ - \int_{\partial_f \Omega^0 \times \mathcal{P} \cap \partial \mathcal{W}_\Pi} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{f}}(\underline{\boldsymbol{X}}, t, \{p\}) \ d\Gamma^0 dp = \int_{\mathcal{W}_\Pi} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v d\Omega^0 dp + \int_{\partial_f \Omega^0 \times \mathcal{P} \cap \partial \mathcal{W}_\Pi} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b d\Gamma^0 dp \quad \forall \underline{\boldsymbol{u}}^* \in \mathcal{W}_\Pi, \ \ \forall t \in]t_0, t_f] \tag{4.11}$$

Si $mes(\Omega^0 \times \mathcal{P})\eta_\Pi > mes(\mathcal{W}_\Pi)C_\epsilon \epsilon_r$, où $C_\epsilon < 1$ (en pratique, $C_\epsilon = 0.1$), cette étape de prévision est suivie par une étape de correction.
Celle-ci consiste à trouver une représentation à variables séparées adaptée satisfaisant l'équation (4.7).
Cette équation étant non-linéaire, en trouver une solution est avantageuse par rapport à l'étape de prévision.
En effet, comme la résolution est itérative, du fait de la non-linéarité, en partant d'une prévision qui approche de la solution finale, il y a moins d'itérations à effectuer pour traiter la non-linéarité. Il s'agit d'une bonne initialisation du processus itératif lié au traitement des non-linéarités.
On trouve cette solution grâce à un algorithme itératif de Newton-Raphson.
On complète la prévision u par plusieurs corrections $\underline{\boldsymbol{\delta_u}}$. Ces corrections sont obtenues en résolvant le problème linéaire en $\underline{\boldsymbol{\delta_u}}$ suivant :

Trouver $\underline{\boldsymbol{R}}_X(\underline{\boldsymbol{X}})$ et $R_p(\{p\})$ tels que :

$$\underline{\boldsymbol{\delta_u}} = \sum_k \underline{\boldsymbol{R}}_{X_k}(\underline{\boldsymbol{X}}) R_{p_k}(\{p\}) \tag{4.12}$$

et

$$\int_{\Omega^0 \times \mathcal{P}} \varepsilon(\underline{\boldsymbol{u}}^*, \underline{\boldsymbol{u}}) \ : \ \boldsymbol{\Sigma}(\underset{\sim}{\boldsymbol{F}}(\underline{\boldsymbol{u}}), \tau \leq t; \{p\}) + \underset{\sim}{\boldsymbol{K}} : \varepsilon(\underline{\boldsymbol{\delta_u}}, 0) \ d\Omega^0 dp - \int_{\partial_f \Omega^0 \times \mathcal{P}} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{f}}(\underline{\boldsymbol{X}}, t, \{p\}) \ d\Gamma^0 dp = \int_{\Omega^0 \times \mathcal{P}} \underline{\boldsymbol{u}}^* . \underline{\hat{\boldsymbol{r}}}_v d\Omega^0 dp + \int_{\partial_f \Omega^0 \times \mathcal{P}} \underline{\boldsymbol{u}}^* . \underline{\hat{\boldsymbol{r}}}_b d\Gamma^0 dp \quad \forall \underline{\boldsymbol{u}}^* \in \mathcal{W}_\Pi, \ \ \forall t \in]t_0, t_f] \tag{4.13}$$

où $\underset{\sim}{\boldsymbol{K}}$ est le tenseur rigidité tangent classique. Comme proposé en [Gonzalez et al., 2010], $\mathcal{V}_R$ est l'espace des fonctions tests définies comme telles :

$$\underline{\boldsymbol{u}}^* = \underline{\boldsymbol{R}}_X^*(\underline{\boldsymbol{X}})R_p(\{p\}) + \underline{\boldsymbol{R}}_X(\underline{\boldsymbol{X}})R_p^*(\{p\}) \tag{4.14}$$

A cause de ce choix de fonctions tests, le problème précédent est un problème non-linéaire en $\underline{\boldsymbol{R}}_X$ et R_p. Les nouvelles fonctions $\underline{\boldsymbol{\Phi}}$ et g sont obtenues en normant respectivement les fonctions $\underline{\boldsymbol{R}}$ et R_p. On obtient une solution approchée satisfaisante des équations (4.12) à (4.13) grâce à l'algorithme du point fixe, en choisissant un critère de qualité relativement souple. Le premier champ $\underline{\boldsymbol{R}}_X$ est donné par les conditions d'équilibres relatives au point de $\mathcal{P}$ ayant le résidu d'équilibre le plus élevé, puis, $\underline{\boldsymbol{R}}_X$ est approximé grâce à la description éléments Finis 3D classique.
R_p est une fonction constante par morceaux.
Ainsi, la simulation multi-dimensionnelle peut être vue comme une simulation simultanée de problèmes mécaniques similaires. La problème à résoudre pour trouver $\underline{\boldsymbol{R}}_X$ a la même dimension que le modèle classique éléments Finis relatif à une unique valeur de paramètres.
A cause de la représentation à variables séparées des déplacements, mais aussi de l'utilisation de la représentation Éléments Finis des fonctions de $\underline{\boldsymbol{X}}$, ce n'est pas toujours possible de choisir ϵ_R aussi petit qu'on le voudrait.

4.2 Présentation de l'algorithme

Rappelons notre objectif : nous voulons traiter des problèmes similaires dans le cadre d'une étude de sensibilité ou d'une surface de réponses.
Dans l'algorithme APHR présenté en section 2.3.1, l'étape 4 consiste à chercher une correction, et ce calcul de la correction Éléments Finis est effectué en utilisant un algorithme de Newton-Raphson. Nous rappelons que la correction des déplacements est telle que :

$$\begin{aligned}\int_{\Omega^0} \varepsilon(\underline{\boldsymbol{u}}^*, \underline{\boldsymbol{u}}_{POD}^{(n)} + \delta\underline{\boldsymbol{u}}_h^{(n)}) \ : \ \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{F}}(\underline{\boldsymbol{u}}_{POD}^{(n)} + \delta\underline{\boldsymbol{u}}_h^{(n)}) \, , \, \tau \leq t; \{p\}_\alpha) \ d\Omega^0 \\ - \int_{\partial_f\Omega^0} \underline{\boldsymbol{u}}^* \cdot \underline{\boldsymbol{f}}(\underline{\boldsymbol{X}}, t; \{p\}_\alpha) \ d\Gamma^0 = \int_{\Omega^0} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v d\Omega^0 + \int_{\partial_f\Omega^0} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b d\Gamma^0 \\ \forall \, \underline{\boldsymbol{u}}^* \in \mathcal{V}_h, \ \ \forall t \in]t_0, t_f]\end{aligned} \tag{4.15}$$

Puisque c'est le coût de cette étape que l'on cherche à diminuer, il est donc nécessaire de diminuer le coût du solveur linéaire.

Nous cherchons donc à programmer un solveur linéaire pour les systèmes matriciels de type bloc sur la diagonale. En effet, les problèmes étant découplés, chaque bloc de la matrice de rigidité correspond à un problème.
Nous obtiendrons ainsi une résolution simultanée de problèmes découplés.

Dans le cadre d'une mise en œuvre d'un algorithme de Newton Raphson, nous devons résoudre des systèmes linéaires de la taille d'un seul cas de calcul. En effet, la dimension du problème à résoudre pour trouver $\underline{\boldsymbol{R}}_X$ est la même que celle d'un problème Éléments Finis relatif à une unique valeur de paramètres.

La matrice de rigidité $[K]$ du problème multidimensionnel s'écrit sous la forme d'une matrice diagonale par bloc, et les vecteurs résidus $\{R\}$ et déplacements $\{u\}$ s'écrivent de la manière suivante :

$$[K] = \begin{pmatrix} \left[\tilde{K}_1\right] & 0 & ... & 0 & 0 \\ 0 & \left[\tilde{K}_2\right] & ... & 0 & 0 \\ 0 & 0 & ... & 0 & 0 \\ 0 & 0 & ... & \left[\tilde{K}_{m-1}\right] & 0 \\ 0 & 0 & ... & 0 & \left[\tilde{K}_m\right] \end{pmatrix} \quad , \quad \{R\} = \begin{pmatrix} \tilde{R}_1 \\ \tilde{R}_2 \\ . \\ . \\ . \\ \tilde{R}_{m-1} \\ \tilde{R}_m \end{pmatrix}$$

$$\{u\} = \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ . \\ . \\ . \\ \tilde{u}_{m-1} \\ \tilde{u}_m \end{pmatrix}$$

où chaque bloc $\left[\tilde{K}\right]$ est associé à un problème relatif à un jeu de paramètres. On considère que l'on a m problèmes à traiter.

L'algorithme du solveur linéaire a été programmé dans le code de calcul ZéBuLoN, et les principales étapes sont données dans l'Algorithme 9.

Data: $[K]$, $\{R\}$, $\{u\}$, m, le vecteur force $\{F\}$, ε_{solve} (seuil de convergence)
Result: la nouvelle valeur des déplacements u
initialisation;
$\{R\} = \{F\} - [K] \cdot \{u\}$;
Recherche de r_{max}, le rang correspondant au résidu $\left\{\tilde{R}\right\}_{r_{max}}$ de norme sup maximum;
$|| \left\{\tilde{R}\right\}_{r_{max}} ||_{ini} = || \left\{\tilde{R}\right\}_{r_{max}} ||$;
while $(|| \left\{\tilde{R}\right\}_{r_{max}} || > \varepsilon_{solve})$ **do**
  Extraire $\left[\tilde{K}\right]_{r_{max}}$ de $[K]$;
  Résolution de $\left[\tilde{K}\right]_{r_{max}} . \{\delta_u\}_{r_{max}} = \left\{\tilde{R}\right\}_{r_{max}}$ à l'aide d'un solveur pour matrices creuses;
  Correction des déplacements correspondant au bloc n^{o} r_{max} (correspondant à chaque fonction de l'espace) :
  $\{\tilde{u}\}_{r_{max}} = \{\tilde{u}\}_{r_{max}} + \{\delta_u\}_{r_{max}}$;
  $\left\{\tilde{B}\right\} = \frac{\{\delta_u\}_{r_{max}}}{||\{\delta_u\}_{r_{max}}||}$;
  Recherche de fonctions dépendantes des paramètres;
  for $i \neq r_{max}$ *tel que le problème* n^{o} i *n'ait pas encore été traité* **do**
    $b_i = \frac{\left\{\tilde{B}\right\}^T \left\{\tilde{R}\right\}_i}{\left\{\tilde{B}\right\}^T \left[\tilde{K}\right]_i \left\{\tilde{B}\right\}}$;
    $\{\delta_u\}_i = \left\{\tilde{B}\right\} b_i$ (produit d'une fonction d'espace et d'une fonction des paramètres);
    Correction des déplacements correspondant au bloc n^{o} i :
    $\{\tilde{u}\}_i = \{\tilde{u}\}_i + \{\delta_u\}_i$;
    Correction des résidus correspondant au bloc n^{o} i :
    $\left\{\tilde{R}\right\}_i = \left\{\tilde{R}\right\}_i - \left[\tilde{K}\right]_i \{\delta_u\}_i$;
  end
  Nouvelle recherche de r_{max};
end

Algorithm 9: Solveur linéaire pour les systèmes matriciels de type bloc sur la diagonale

4.3 Construction d'un RID multidimensionnel

Dans la section 2.4, nous avons décrit les principales étapes de la construction d'un RID.
Nous reprenons ici les étapes de cette construction en simulation multidimensionnelle : seules les équations (2.52) et (2.53) ontété modifiées.
Au début de ce chapitre, nous avons défini $\tilde{\Omega} = \Omega \times \mathcal{P}$ comme l'espace du maillage dupliqué (cf. FIG. 4.2). Cet espace est tel que $\tilde{\Omega} = \bigcup_{e_i=1}^{N_e} \tilde{\Omega}_{e_i}$.

- Création d'un premier set de s éléments assurant la "visibilité" du chargement extérieur par notre formulation et constituant le RID initial $\mathcal{L}_g^{(0)}$.
- Ajout des régions d'intérêt particulier.
- Ajout des régions où se passent les transformations les plus significatives. Les vecteurs de la base $(\underline{\boldsymbol{Y}}_k)_{k=1\ldots\xi}$ représentant ces transformations pour les variables internes, nous choisirons donc les ξ éléments où l'énergie associée aux variables internes atteint son maximum, *i.e* :

$$e_j = \arg\max_{e_i} \int_{\tilde{\Omega}_{e_i}} \underline{\boldsymbol{Y}}_k \cdot \underline{\boldsymbol{Y}}_k d\Omega \tag{4.16}$$

  Nous obtenons ainsi la version $\mathcal{L}_g^{(1)}$ du RID.
- Nous pouvons aussi choisir pour chaque vecteur $\underline{\boldsymbol{\psi}}_k$ l'élément où se localise le maximum de la déformation, soit :

$$e_j = \arg\max_{e_i} \int_{\tilde{\Omega}_{e_i}} \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{\psi}}_k) \cdot \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{\psi}}_k) d\Omega \tag{4.17}$$

  Nous choisirons alors les d éléments supplémentaires et ceci constitue la version $\mathcal{L}_g^{(2)}$ du RID.

L'extension du RID par ajout des éléments connectés à ceux déjà sélectionnés constitue la version finale $\mathcal{L}_g^{(3)}$ du RID.

4.4 Résultats numériques d'une simulation multidimensionnelle

4.4.1 Analyse de sensibilité.

Tous les résultats présentés dans cette section ont été publiés dans [Sarbandi et al., 2010]. Le lecteur pourra trouver cet article en annexe 4.5. Nous avons précisé lors de l'introduction que l'état mécanique d'un système peut être vu comme une fonction dépendant non seulement du temps et des variables de l'espace, mais aussi des coefficients matériau, dans le cadre d'une simulation multidimensionnelle.
La simulation multidimensionnelle des matériaux semble être une approche séduisante grâce aux récents travaux d'Ammar et al. [Ammar et al., 2006b] [Ammar et al., 2006a] sur les algorithmes basés sur la représentation en variables séparées. Les modèles que nous étudions sont multidimensionnels, dû à la variation des propriétés des matériaux.
L'algorithme proposé, ainsi que la méthode de séparation de variables et la méthode d'intégration tronquée sont génériques. Ils peuvent être appliqués à différents problèmes mécaniques. Pour illustrer le fait que l'on peut appliquer la méthode proposée à des lois de comportement très complexes, nous allons appliquer notre méthode afin de simuler des transformations de frittage.
Les paramètres matériaux du modèle constitutif doivent être déterminés expérimentalement, et utilisés lors de l'analyse Éléments Finis.
Il est évident que, si les résultats de la simulation Éléments Finis de la déformation par frittage peuvent être fortement affectés par les variations de certains paramètres, l'effet de ces mêmes variations sur d'autres paramètres peut être négligeable.
Par conséquent, l'analyse de l'effet d'une variation de paramètres est importante, car elle peut fournir une aide sur le niveau d'incertitude que l'on peut tolérer pour ces paramètres. En d'autres termes, les paramètres ayant peu d'effet peuvent être identifiés pour une classe de matériaux assez large. Mais ceux dont l'effet produit est important, doivent être spécifiés avec une grande précision pour chaque matériau considéré.

4.4.2 Loi constitutive pour le frittage.

Les transformations de frittage peuvent être vues comme des transformations thermomécaniques avec déformations thermiques irréversibles couplées à des transformations viscoplastiques. Le modèle utilisé ici a été développé dans le cadre de la thèse de Sarbandi [Sarbandi et al., 2010]. On suppose

que la température est uniforme sur tout le domaine multidimensionnel $\Omega^0 \times \mathcal{P} \times]t_0, t_f]$. En général, le taux de déformation pendant le cycle de frittage d'un corps céramique est constitué de trois termes : le taux de déformation thermique réversible, le taux de déformation durant le frittage irréversible, et le taux de déformation viscoplastique, comme le montrent les équations suivantes.

$$\underset{\sim}{\dot{\boldsymbol{\varepsilon}}} = \dot{\varepsilon}_{th}\, \underset{\sim}{\mathbf{1}} + \dot{\varepsilon}_s(f,T)\, \underset{\sim}{\mathbf{1}} + \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_{vp} \tag{4.18}$$

où $\underset{\sim}{\mathbf{1}}$ est le tenseur unité du second ordre. Le tenseur taux de déformation est défini en utilisant la décomposition polaire du gradient de déformation :

$$\underset{\sim}{\boldsymbol{F}} = \underset{\sim}{\boldsymbol{R}}.\underset{\sim}{\boldsymbol{U}} \quad \text{and} \quad \underset{\sim}{\dot{\boldsymbol{\varepsilon}}} = \frac{1}{2}\, \underset{\sim}{\boldsymbol{R}}^T.(\underset{\sim}{\dot{\boldsymbol{F}}}.\underset{\sim}{\boldsymbol{F}}^{-1} + \underset{\sim}{\boldsymbol{F}}^{-T}.\underset{\sim}{\dot{\boldsymbol{F}}}^T).\underset{\sim}{\boldsymbol{R}} \tag{4.19}$$

où $\underset{\sim}{\boldsymbol{R}}$ est un tenseur de rotation pure. La contrainte conjuguée $\underset{\sim}{\boldsymbol{G}}$ est telle que :

$$\underset{\sim}{\boldsymbol{\sigma}} = \underset{\sim}{\boldsymbol{R}}.\underset{\sim}{\boldsymbol{G}}.\underset{\sim}{\boldsymbol{R}}^T \tag{4.20}$$

Le taux de déformation durant le frittage est une fonction de la température (T) et de la porosité (f) et s'écrit :

$$\dot{\varepsilon}_s(f,T) = -f^y A_s \exp(\frac{-Q_s}{R\,T}) \tag{4.21}$$

Le taux de déformation viscoplastique induit par la contrainte mécanique $\underset{\sim}{\boldsymbol{G}}$ s'écrit :

$$\underset{\sim}{\boldsymbol{G}} = \eta(T)\, \underset{\sim}{\boldsymbol{M}}^{(4)}(T,f) : \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_{vp} \tag{4.22}$$

où η est la viscosité $\underset{\sim}{\boldsymbol{M}}^{(4)}(T,f)$ est le tenseur du quatrième ordre, qui, en général, ne dépend pas du tenseur des contraintes $\underset{\sim}{\boldsymbol{\sigma}}$ quand le niveau de contrainte est très petit :

$$\underset{\sim}{\boldsymbol{M}}^{(4)}(T,f) = F(f)\, \underset{\sim}{\boldsymbol{I}}^{(4)} + \frac{3}{2} C(f)\, \underset{\sim}{\boldsymbol{J}}^{(4)} \tag{4.23}$$

où $\underset{\sim}{\boldsymbol{I}}^{(4)}$ et $\underset{\sim}{\boldsymbol{J}}^{(4)}$ sont des tenseurs du quatrième ordre :

$$\underset{\sim}{\boldsymbol{I}}^{(4)} : \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_{vp} = Tr(\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_{vp})\, \underset{\sim}{\mathbf{1}} \tag{4.24}$$

$$\underset{\sim}{\boldsymbol{J}}^{(4)} : \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_{vp} = \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_{vp} - \frac{1}{3}\, Tr(\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}_{vp})\, \underset{\sim}{\mathbf{1}} \tag{4.25}$$

Les coefficients C et F dépendent de la porosité f (dans le cas d'un corps dense, $f = 0$, $C = 1$ et $F = 0$). On suppose qu'ils sont de la forme suivante :

$$C(f) = 1 + x\, f^n \quad F(f) = z\, f^m \tag{4.26}$$

La viscosité s'écrit :

$$\eta(T) = B_s \exp(\frac{-Q_s}{RT}) \tag{4.27}$$

où Q_s et T sont respectivement l'énergie d'activation et la température absolue. Le changement de porosité est gouverné par le taux de déformation tel que :

$$\dot{f} = (1-f)\, Tr(\dot{\underset{\sim}{\varepsilon}} - \dot{\varepsilon}_{th}\, \underset{\sim}{\mathbf{1}}) \tag{4.28}$$

A_s, B_s, Q_s, m, n, x, y et z sont les coefficients intrinsèques du matériau, qui doivent être identifiés par des tests expérimentaux.

4.4.3 La représentation Éléments Finis

Comme nous l'avons précisé dans la section précédente, les déformations istropiques dues au frittage sont constituées de plusieurs paramètres associés au frittage (thermique) ou aux sollicitations mécaniques. De par la variabilité des propriétés du matériau, il peut s'avérer intéressant d'étudier la sensibilité de la réponse mécanique à différents coefficients matériau. Les paramètres d'un modèle doivent être déterminés expérimentalement, et utilisés lors de l'analyse Éléments Finis. Il est évident que, tandis que les résultats de la simulation Éléments Finis d'un certain modèle seront fortement affectés par les variations de certains paramètres, l'effet de ces mêmes variations sur d'autres paramètres sera négligeable.
Nous nous intéressons à 2 réponses de l'effet de la perturbation des paramètres mécaniques sur la flexion d'une poutre :
(i) le déplacement horizontal maximum de la poutre encastrée durant le traitement thermique (U_1)
(ii) la flexion maximum de la poutre encastrée due au frittage (U_2).
Ces 2 réponses sont linéairement reliées aux 5 paramètres par une matrice de sensibilité $[S]$.

La colonne des réponses est notée $\{Y\}$, $\{Y\}^T = \{U_2, U_1\}$. On considère deux niveaux pour chacun des cinq paramètres mécaniques du modèle.

En d'autres termes, 2^5 points de $\mathcal{P}$ sont examinés afin de créer un plan factoriel complet. Nous proposons de simuler ces 32 cas numériques de manière simultanée. La représentation des paramètres étant constante par morceaux, il s'avère que le modèle numérique est un modèle Éléments Finis discontinu, défini dans un espace euclidien $\widehat{\Omega}^0$ qui imite le domaine $\Omega^0 \times \mathcal{P}$. Le système mécanique défini dans $\widehat{\Omega}^0$ contient les 32 poutres qui sont associées à chaque cas d'étude (pour chaque perturbation). Cette géométrie est présentée en figure FIG. 4.3. Ce maillage est assez lourd, puisque chaque problème contient 400 éléments. La maillage entier est donc constitué de 12800 élements quadratiques (32×400).

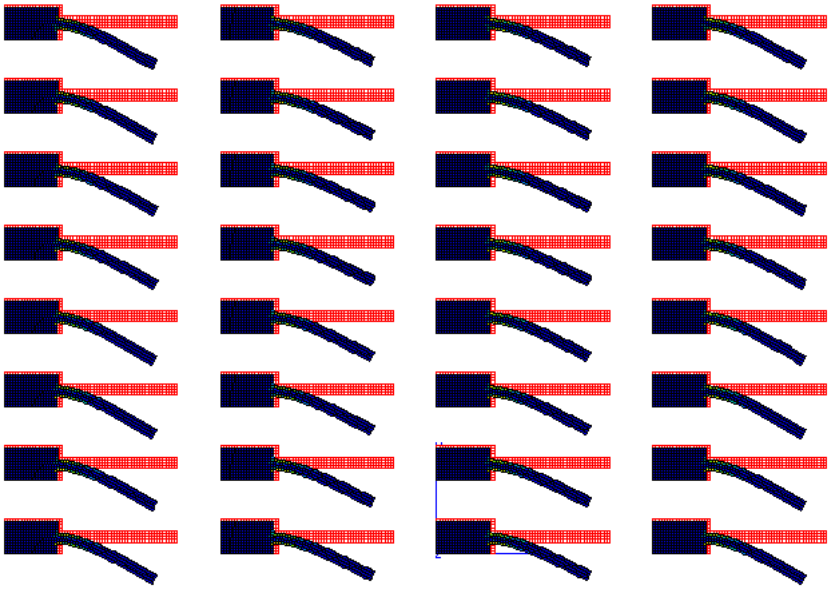


FIG. 4.3 – Maillage et exemple de la contrainte de Von Mises relative au problème multidimensionnel vu dans l'espace virtuel 2D $\widehat{\Omega}^0$ qui imite $\Omega^0 \times \mathcal{P}$ [Sarbandi et al., 2010]p.

4.4.4 Assignation des paramètres

Les paramètres dont les valeurs sont constantes tout au long de l'étude, sont tous associés à l'aspect frittage (thermique) du modèle. Ces paramètres sont fixés et leurs valeurs sont listées en table TAB. 4.1.

Paramètres du modèle	Valeurs optimisées
A_S	1.2548274354e+3
y	4.5206134869
Q_S	1.8133327934e+5

TAB. 4.1 – Paramètres de frittage après optimisation.

Nous avons vu que la loi de comportement dépend, entre autres, de 5 coefficients matériaux B_S,m,n,x,z qui sont obtenus par résolution d'un problème inverse. Les valeurs obtenues de ces coefficients sont présentées dans la table TAB. 4.2.

Les deux premiers paramètres (x et n) sont liés à la partie déviatorique du modèle qui représente la déformation due au cisaillement pendant le frittage. Les deux paramètres suivants (z et m) sont liés à la partie sphérique du

Paramètres du modèle	Valeurs optimisées
x	1.45386139054
n	1.06450577625
z	9.16276723203e-1
m	1.2735633572
B_S	8.76336201142e-3

TAB. 4.2 – Paramètres mécaniques après optimisation.

modèle. Ils représentent la déformation volumique (rétrécissement due au frittage) de la céramique. Le dernier paramètre influent B_S est représentatif de la viscosité de la pièce en céramique pendant le traitement thermique.

On regarde l'influence d'une perturbation de $\pm 5\%$ de ces paramètres mécaniques sur le modèle. On a donc 2 niveaux par paramètre, soit $2^5 = 32$ cas numériques.

4.4.5 Construction de la matrice de sensibilité.

Nous avons assigné à chacun des cinq paramètres mécaniques du modèle une valeur "haute" et une valeur "basse", et nous allons effectuer une unique simulation contenant toutes les combinaisons possibles de ces valeurs de paramètres, soit 32 problèmes différents.

$$\{p\} = \begin{pmatrix} x \\ n \\ z \\ m \\ B_s \end{pmatrix} = \{p_0\}, \{p_1\}, \{p_2\}...\{p_{32}\} \tag{4.29}$$

À la fin de cette simulation simultanée, on utilise des techniques standards pour analyser la variation de la réponse des variables, grâce au tableur Excel et au logiciel de calcul matriciel Matlab.

On suppose que les deux réponses concernées U_1 et U_2 sont linéairement reliées aux cinq paramètres par une matrice de sensibilité $[S]$. La différence de la valeur des réponses par rapport au cas de référence a été calculée pour tous les cas possibles de la manière suivante :

$$\{\delta Y_i\} = \{Y_i\} - \{Y_0\} \tag{4.30}$$

De la même manière, la variation de la valeur des coefficients par rapport

à leur valeur de référence est déterminée de cette façon :

$$\{\delta p_i\} = \{p_i\} - \{p_0\} \tag{4.31}$$

Alors la matrice de sensibilité est décomposée en deux matrices différentes, et est reliée à la matrice de déviation des paramètres telle que :

$$\{\delta Y_i\} = [G_i].\{\widehat{S}\} = [S].\{\delta p_i\} \tag{4.32}$$

En minimisant le terme suivant, on obtient $d(\widehat{S})$, qui est utilisé comme indice de sensibilité.

$$J(\widehat{S}) = \sum_{i=1}^{32}(\{\delta Y_i\} - [G_i].\{\widehat{S}\})^T.(\{\delta Y_i\} - [G_i].\{\widehat{S}\}) \tag{4.33}$$

$$dJ(\widehat{S}) = -2\sum_{i=1}^{32} d(\widehat{S})^T.[G_i]^T(\{\delta Y_i\} - [G_i].\{\widehat{S}\}) \tag{4.34}$$

$$dJ(\widehat{S}) = 0 \tag{4.35}$$

$$(\sum_{i=1}^{32}[G_i]^T.[G_i]).\widehat{S} = \sum_{i=1}^{32}[G_i]^T.\{\delta Y_i\} \tag{4.36}$$

4.4.6 Comparaison des résultats obtenus avec la méthode EF et la méthode APHR

Dans cette section, nous allons comparer les résultats fournis par la méthode APHR proposée avec les résultats obtenus par une simulation classique séquentielle Éléments Finis. En effet, nous choisissons de prendre comme référence un calcul séquentiel Éléments Finis standard, car les problèmes étant découplés, cela n'aurait pas de sens de traiter ce problème par une simulation simultanée. Cette solution Éléments Finis classique consiste à traiter chaque problème correspondant à un set de paramètres l'un après l'autre.

Nous pouvons trouver en figure FIG. 4.4 l'histogramme de sensibilité des simulations Éléments Finis. Ce sont les 10 coefficients (5 paramètres $\times$ 2 réponses) de la matrice de sensibilité $[S]$ qui sont représentés. L'axe x représente les 5 paramètres $P_{i,(i=1,\ldots,5)}$ (respectivement x, n, z, m, B_S) tandis que les réponses (U_1 et U_2) peuvent être localisées sur l'axe y.

Sur l'axe z, nous pouvons trouver l'intensité des coefficients de la matrice de sensibilité $[S]$ en échelle logarithmique.

Nous pouvons alors remarquer que le paramètre le plus sensible à une perturbation de $\pm 5\%$ est le cinquième paramètre, B_S (paramètre relié à la viscosité), tandis qu'une même perturbation des paramètres liés à la déformation volumique due aux effets mécaniques (z et m) ne perturbe aucune réponse.

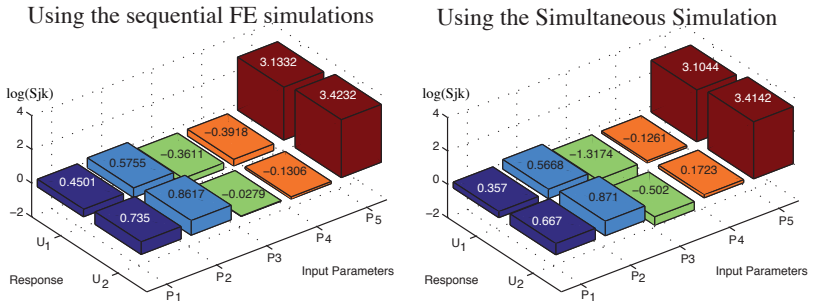


FIG. 4.4 – Sensibilité de la réponse à une perturbation de paramètres pour la simulation séquentielle Éléments Finis [Sarbandi et al., 2010].

De ce fait, la flexion de la poutre encastrée et le déplacement horizontal qu'elle subit pendant le frittage sont plus sensibles à la perturbation visqueuse, plutôt qu'à la partie déviatorique ou sphérique du modèle. Ainsi, le paramètre de viscosité doit être déterminé précisément. Les autres paramètres mécaniques du modèle peuvent être déterminés pour une classe plus large de matériaux similaires.

Les résultats quantitatifs obtenus en utilisant la méthode APHR sont très proches de ceux obtenus avec la méthode Éléments Finis (FIG. 4.4) : le paramètre le plus sensible est la paramètre de viscosité (B_S), et aucune des réponses n'est perturbée par une variation des paramètres reliés à la partie sphérique du modèle (z et m).

4.4.7 Efficacité de la simulation simultanée en utilisant la méthode APHR.

Dans cette partie, nous comparererons trois stratégies de calcul :

- la simulation classique éléments Finis,
- la simulation simultanée de problèmes en utilisant la méthode APR,
- la simulation simultanée de problèmes en utilisant la méthode APHR.

Comme nous l'avons déjà mentionné, le problème à résoudre pour trouver $\underline{\boldsymbol{R}}_X$ est de même dimension que le modèle classique Éléments Finis relatif à une unique valeur de paramètres. Pour chaque stratégie, nous pourrons comparer :

- le nombre N_g de problèmes linéaires classiques Éléments Finis,

- le nombre N_l de calculs locaux de la contrainte aux points de Gauss, sur l'incrément de temps concerné (afin de vérifier que le problème a bien été simplifié, suite à l'utilisation de l'Hyper Réduction.),
- le nombre N_u d'inconnues du système.

Le Domaine d'Intégration Réduit a été construit en ajoutant tous les éléments du premier maillage comme zone d'intérêt. La figure FIG. 4.5 montre les éléments en rouge sur lesquels la fonction h dans l'équation (4.9) n'est pas nulle. Ce maillage contient en moyenne 750 éléments, soit 17 fois moins que dans le maillage entier.

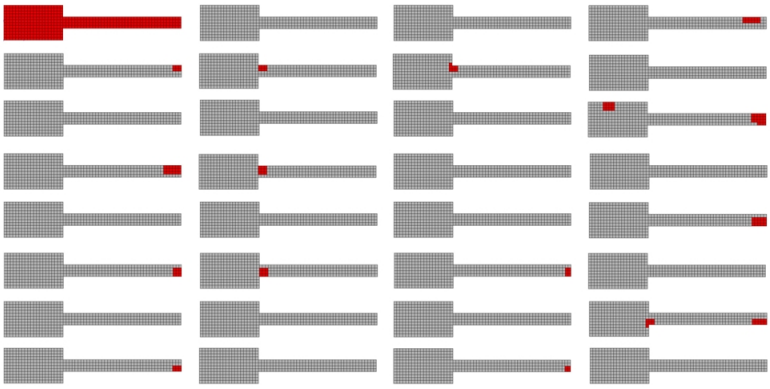


FIG. 4.5 – Maillage Éléments Finis et Domaine d'Intégration Réduit (en rouge) utilisés pour l'analyse de sensibilité des paramètres mécaniques avec la méthode EF ou les méthodes de réduction a priori [Sarbandi et al., 2010].

Les paramètres de la méthode de réduction de modèles sont $\epsilon_R = 3.10^{-3}$ et $\varepsilon_{POD} = 10^{-8}$.

Le tableau TAB. 4.3 résume les résultats obtenus.

Pour la stratégie Éléments Finis classique, N_u est le nombre de degrés de libertés du système, soit 2546 (comme les problèmes simultanés ne sont pas couplés, cela n'aurait aucun sens de comparer le nombre de degrés de libertés (32×2546) du modèle Éléments Finis simultané.). Pour les stratégies utilisant la méthode APHR, il s'agit du nombre de modes empiriques contenus dans la base, soit 5 modes. La POD permet de réduire le nombre d'inconnues N_u. Celui-ci a été divisé par 510.

Pour la stratégie proposée, N_l est la valeur définie dans l'équation (4.4) à la fin de la simulation simultanée. Pour la stratégie Éléments Finis, il s'agit du nombre d'appels de résolutions des problèmes locaux, multiplié par le nombre de points de gauss du système (4 points de gauss pour chaque élément), soit $4 \times 12800 \times 1174 = 60,108,800$. Pour la stratégie APR ainsi que la méthode

Stratégies	N_g	N_l	N_u
EF séquentiel	$32 \times 981 = 31392$	$12800 \times 4 \times 1183 = 60,569,600$	2546
Simulation simultanée sans HR (APR)	1399	$12800 \times 4 \times 1174 = 60,108,800$	5
Simulation simultanée avec HR (APHR)	1399	$12800 \times 4 \times 72 + 750 \times 4 \times 1358 = 7,760,400$	5

TAB. 4.3 – Nombre de solutions globales N_g, de solutions locales N_l et nombre d'inconnues N_u relatifs à chaque stratégie

APHR, il s'agit de la somme du nombre d'appels de résolutions de problèmes locaux pendant l'étape de prédiction (dans cette étape, qui se fait en modèle d'ordre réduit, le nombre d'éléments est celui contenu le domaine entier pour l'APR, soit 12800, et dans le RID pour l'APHR, soit 750 en moyenne) , et pendant l'étape de correction (cette étape se faisant en base complète, le nombre d'éléments est le nombre total d'éléments, soit 12800). L'utilisation de l'Hyper Réduction a permis de diviser ce nombre par 6.5.

On rappelle que la dimension du problème à résoudre pour trouver R_x est la même que celle d'un modèle Éléments Finis relatif à une unique valeur de paramètres. C'est pour cela que l'on comparera le nombre de problèmes linéaires Éléments Finis à résoudre pour chaque stratégie. Ce nombre est noté N_g. Pour la stratégie Éléments Finis, il s'agit du nombre total d'itérations multiplié par le nombre de problèmes, soit $981 \times 32 = 31392$. Pour les deux autres méthodes, il s'agit du nombre d'appels à la résolution des problèmes linéaires globaux, c'est-à-dire le nombre de passages dans le solveur linéaire décrit dans la section 4.2. Le nombre de problèmes linéaires globaux pour la méthode Éléments Finis a été divisé par 22, prouvant ainsi l'efficacité de la méthode proposée.

L'utilisation de la méthode APHR rend donc le solveur basé sur la représentation à variables séparées très efficace, dans le cas de simulations simultanées impliquant des équations inséparables.

4.4.8 Conclusion.

L'analyse sensitive des paramètres mécaniques d'un modèle de frittage a été effectuée suivant trois différentes stratégies :

- la simulation classique séquentielle Éléments Finis,
- la simulation simultanée de problèmes en utilisant la méthode APR,
- la simulation simultanée de problèmes en utilisant la méthode APHR.

La méthode de variables séparées a été appliquée en distinguant trois sortes de fonctions :

- les fonctions de l'espace,
- les fonctions du temps,
- les fonctions dépendant des coefficients matériau du modèle de frittage.

Il est clair que l'Hyper Réduction améliore l'efficacité de la méthode des variables séparées dans un cas impliquant des équations inséparables. Lors de la construction de la méthode des variables séparées, de nouvelles fonctions d'états ont été construites, satisfaisant le système linéaire de dimension égale à la dimension du modèle Éléments Finis.
La réduction de la complexité du problème est due à la réduction du nombre de problèmes linéaires à résoudre. En effet, ce nombre a été divisé par 22.

4.5 Autre exemple numérique de simulation multidimensionnelle "fil conducteur"

Reprenons le problème inverse présenté en section 3.7.2 dont nous rappelons le maillage et les conditions aux limites ci-dessous (FIG. 4.6).

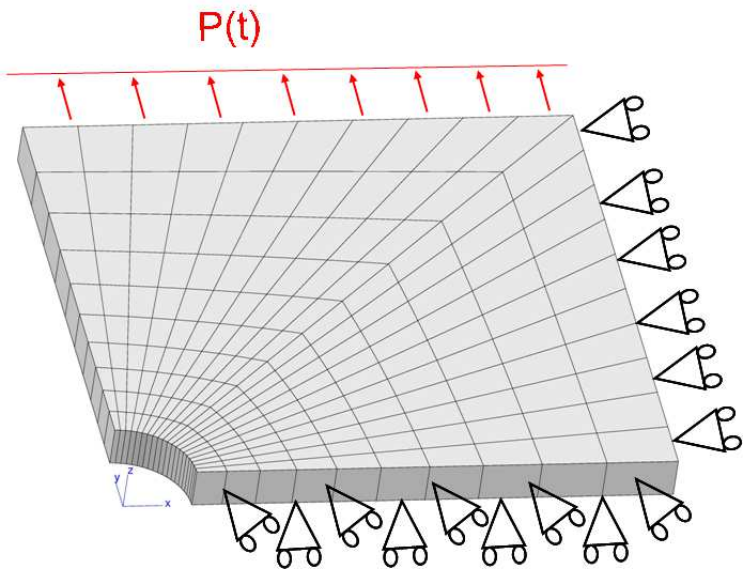


FIG. 4.6 – Maillage d'un huitième de l'éprouvette et conditions aux limites.

Avant d'atteindre les valeurs optimisées, l'optimiseur Simplex de Zébulon a effectué une suite de simulations différants les unes des autres par une variation des paramètres. Nous avons vu qu'il a fallu 122 itérations avant d'arriver à convergence (soit 122 simulations effectuées). Nous décidons de garder les 32 dernières simulations effectuées, et d'enregistrer les set de paramèrres correspondant à ces simulations. C'est cette suite de 32 dernières simulations que nous allons recommencer, mais cette fois ci, en effectuant une unique simulation simultanée de ces 32 cas de calculs.

De la même manière que pour l'exemple précédent, afin d'effectuer une unique simulation représentant ces 32 cas de calculs, nous avons besoin d'un maillage constitué des 32 géométries, soit $32 \times 200 = 6400$ éléments (FIG. 4.7).

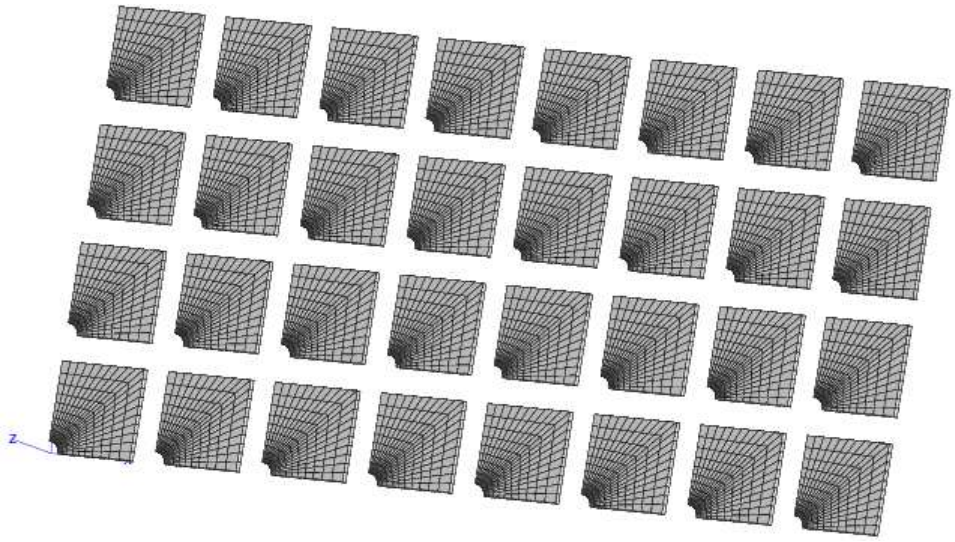


FIG. 4.7 – Maillage Éléments Finis utilisé pour la comparaison de l'efficacité des méthodes EF ou ROM.

Le RID, construit par la méthode d'Hyper Réduction multidimensionnelle, est constitué des éléments apparaissant sur la figure FIG. 4.8 en rouge. En moyenne, il est constitué de 114 éléments, au lieu des 6400 contenus dans le maillage entier.

Une pression $P(t) = 22.08t$ est imposée sur toute la partie supérieure de l'éprouvette.

Les paramètres de la méthode de réduction de modèles sont $\epsilon_R = 5.10^{-3}$ et $\varepsilon_{POD} = 10^{-8}$.

Les résultats sont résumés dans le tableau suivant (TAB. 4.4).

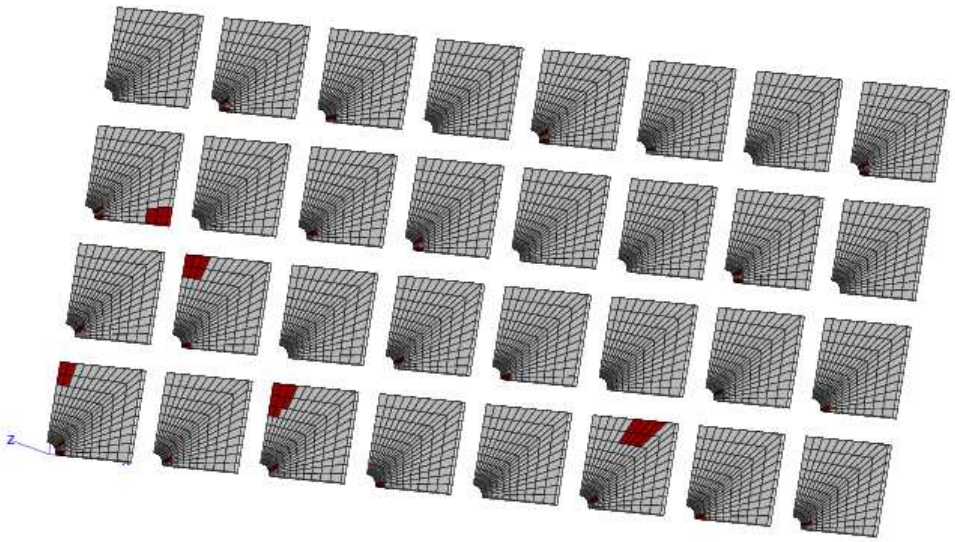


FIG. 4.8 – Domaine d'Intégration Réduit (en rouge) utilisé pour la comparaison de l'efficacité des méthodes EF ou ROM.

Stratégies	N_g	N_l	N_u
EF séquentiel	$32\times202 = 6464$	$6400 \times 4 \times 294 = 7526400$	4659
Simulation simultanée sans HR (APR)	1542	$6400\times4\times54+6400\times4\times222 = 7065600$	18
Simulation simultanée avec HR (APHR)	3002	$6400 \times 4 \times 110 + 114 \times 4 \times 99 = 2861144$	16

TAB. 4.4 – Nombre de solutions globales N_g, de solutions locales N_l et nombre d'inconnues N_u relatifs à chaque stratégie

De même, la POD a permis de réduire le nombre d'inconnues N_u. Celui-ci a été divisé par 259 grâce à l'utilisation de la méthode APR, et par 291 grâce à l'Hyper Réduction. L'utilisation de l'Hyper Réduction a permis de diviser le nombre de résolutions locales de la contrainte aux points de gauss par 2.6. Le nombre de problèmes linéaires globaux pour la méthode Éléments Finis a été divisé par 4 et par 2, suite à l'utilisation respectivement des méthodes APR et APHR, prouvant une fois de plus l'efficacité des méthodes proposées.

Conclusion

Rappelons que le principal objectif de cette thèse était de proposer une méthode de simulation bien adaptée aux problèmes d'optimisation, que ceux-ci soient du type résolution de problèmes inverses (résolution séquentielle de problèmes similaires) ou construction d'une base de données pour construire des surfaces de réponse (résolution simultanée de problèmes similaires).

Le coût de la résolution numérique des problèmes non linéaires peut rapidement devenir prohibitif, plus précisément pour les problématiques industrielles complexes introduisant des modèles Éléments Finis de plusieurs centaines de milliers de degrés de liberté. Ainsi, si on a le besoin de multiplier les simulations (comme dans le cas des problèmes d'optimisation que nous proposons), l'utilisation des calculs par Éléments Finis classiques peut s'avérer assez lourde. C'est pourquoi les méthodes de réduction de modèles ont été introduites. Pour remédier à ce problème. Elles visent à réduire le coût global pour l'obtention d'une solution en recherchant celle-ci dans l'espace d'une base adaptée au problème plutôt que dans l'espace des fonctions de forme Éléments Finis.

Ces deux approches que nous proposons ont donc été traitées à l'aide d'une méthode de réduction de modèles.
Les méthodes utilisées au cours de la thèse (APR ou APHR) ont la particularité d'être :

- basée sur une technique POD pour construire la base réduite : la base POD obtenue est ainsi optimale d'un point de vue énergétique, mais incapable de représenter une information qui n'était pas contenue initialement dans la base de donnée utilisée pour la déterminer.
- adaptative : pour justement pouvoir représenter une nouvelle information.
- a priori : elles ne nécessitent pas de connaissance au préalable de réponse du système physique.
- des méthodes de représentation des déplacements multi-niveau : l'ap-

proche multi-niveau proposée nécessite la définition de deux sous-espaces définissant deux niveaux d'approximation des déplacements.

Mais le coût de la résolution d'un problème non linéaire n'est pas seulement dû à la résolution de systèmes d'équations, l'intégration locale des lois de comportement peut représenter jusqu'à 80% des coûts pour certaines lois de comportement compliquées. Dans ce cas, l'approche APR n'apporte que très peu de gain. Pour améliorer la performance la méthode APHR permet en plus d'effectuer un calcul sur une partie de la structure seulement, réduisant ainsi drastiquement le coût de calcul.

Les méthodes proposées ont étés mises en œuvre sur des problèmes complexes et variés : elles ont été appliquées à différents types de comportements mécaniques : élastoplastique ou plasticité endommageable.

Dans un premier temps, nous avons remarqué que l'utilisation de méthodes de réduction de modèles pour résoudre des suites de simulations similaires au cours du traitement de suites de calculs, ou de problèmes inverses peut conduire à une dégradation des connaissances acquises. En effet, une base est construite pour essayer de représenter les évènements significatifs contenus dans l'ensemble des simulations. Mais l'accumulation de simulations similaires implique que des évènements soient présents dans chacune de ces simulations, et masquent ainsi les évènements spécifiques à certaines simulations.
Nous avons traité des problèmes d'optimisation contenant de nombreuses itérations, et donc susceptibles d'être soumis aux problèmes de ces évènements que nous avons appelé évènements récurrents.
Nous avons pu prouver qu'en effet, en adaptant la base, et à partir d'un certains nombre de simulations réalisées, la qualité du calcul se dégrade. Nous avons donc apporté une solution efficace à la gestion de ces évènements récurrents en développant un algorithme de réduction adaptative dédié aux problèmes d'optimisation avec prise en compte de la présence d'évènements récurrents : il s'agit d'oublier une partie des simulations effectuées en introduisant un facteur d'oubli, choisi par l'utilisateur.
Cette solution s'est avérée efficace, mais nous pouvons déplorer le fait que la recherche de ce facteur d'oubli n'est pas automatique, et doit s'effectuer au cas par cas. Toutefois, nous avons pu observer qu'à chaque cas traité, le facteur d'oubli nous permettant des résultats optimaux est le même ($\gamma = 0.5$). Évidemment, cela ne garantit pas une généralité, mais il nous semble être un bon compromis entre qualité des résultats et rapidité d'obtention de ces résultats. C'est donc cette valeur de facteur d'oubli que nous préconisons et

c'est celle qui a été implémentée par défaut dans le code de calcul ZéBuLoN. Le principal étant d'éviter les deux valeurs de γ ayant des conséquences particulières : $\gamma = 1$ permet de conserver une base valable pour tous les cas de figure, mais qui pose le problème des évènements récurrents et $\gamma = 0$ fournit une base ne conservant que le dernier cas de calcul, au risque de perdre tous les resultats antérieurs.

Nous nous sommes intéressés par la suite au développement d'une méthode numérique de représentation à variables séparées pour la représentation de problèmes paramétriques. En effet les problèmes que nous traitons sont paramétrés, et il ne paraît pas absurde de considérer que le paramètre est une coordonnée à juste titre, comme la variable d'espace ou de temps. Ceci nous a conduit à nous intéresser aux problèmes multidimensionnels, que nous avons décidé de résoudre de manière simultanée, puisqu'une simulation multidimensionnelle peut être vue comme une simulation simultanée de problèmes mécaniques similaires. Pour ce type d'approche, il n'y a pas de problème d'évènements récurrents. Les problèmes que nous traitons étant non-linéaires, il est quelquefois impossible de séparer les opérateurs en temps, en espace et en paramètres. C'est pourquoi nous avons proposé de compléter la méthode APHR pour les méthodes de séparation de variables.
L'utilisation de la méthode APHR a rendu donc le solveur basé sur la représentation à variables séparées très efficace, dans le cas de simulations simultanées impliquant des équations inséparables.
En effet, l'utilisation de la POD permet dans un premier temps de réduire considérablement le nombre de degrés de libertés Éléments Finis du système. Mais l'utilisation de la méthode APHR a permis en plus de réduire le nombre d'appels de résolutions de problèmes locaux réduits à chaque point de Gauss. En effet, l'Hyper Réduction permet de construire un domaine d'intégration réduit, en limitant le nombre d'éléments sur lequel l'intégration locale des lois de comportement s'effectue.
Enfin, le fait de résoudre les problèmes simultanément a permis de réduire le nombre d'appels à la résolution des problèmes linéaires globaux.

Concernant la suite à donner à ces travaux, une des perspectives intéressante serait de développer l'aspect "interpolation de bases". En effet, en utilisant des simulations simultanées en base réduite, nous savons créer une base de données qui permet de construire une surface de réponses. Mais il faut être capable de pouvoir rajouter des points, et pour cela, il faut utiliser des techniques d'interpolation. Notre idée serait de s'inspirer des travaux d'interpolation de bases [Amsallem and Farhat, 2008] pour compléter les tra-

vaux que nous avons réalisé dans notre approche multidimensionnelle.
Nous pourrons également nous intéresser à l'utilisation de la méthode APHR pour des problèmes de mécanique des fluides. Cependant, pour des problèmes de mécanique des fluides, la formulation Petrov-Galerkin à utiliser est particulière.

Table des figures

Liste des tableaux

Bibliographie

[Alexandrov et al., 1998] Alexandrov, N., Dennis, J., Lewis, R., and Torczon, V. (1998). A trust region framework for managing the use of approwimation models in optimization. *Structural Optimization*, 15 :16–23.

[Allix and Vidal, 2002] Allix, O. and Vidal, P. (2002). A new multi-solution approach suitable for structural identification problems. *Comp. Meth. Appl. Mech. Engng.*, 191 :2727–2758.

[Ammar et al., 2006a] Ammar, A., Mokdad, B., Chinesta, F., and Keunings, R. (2006a). A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *J. of Non-Newtonian Fluid Mech.*, 139(3) :153–176.

[Ammar et al., 2006b] Ammar, A., Mokdad, B., Chinesta, F., and Keunings, R. (2006b). A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. part ii : transient simulation using space-time separated representations. *J. of Non-Newtonian Fluid Mech.*, 144 :98–121.

[Ammar et al., 2006c] Ammar, A., Ryckelynck, D., Chinesta, F., and Keunings, R. (2006c). On the reduction of kinetic theory models related to finitely extensible dumbells. *J. Non-Newton. Fluid Mech.*, 134(1).

[Amsallem and Farhat, 2008] Amsallem, D. and Farhat, C. (2008). Interpolation method for adapting reduced-order models and application to aeroelasticity. *American Institute of Aeronautics and Astronautics Journal*, 46 :1803–1813.

[Arzt, 1994] Arzt, M. (1994). *Analyse sous chargement cyclique de structures vicoplastiques*. PhD thesis, ENS Cachan.

[Bai, 2002] Bai, Z. (2002). Krylov subspace techniques for reduced-order modeling of large scale dynamical systems. *Appl. numer. Math.*, 43 :9–44.

[Balima et al., 2006] Balima, O., Favennec, Y., Girault, M., and Petit, D. (2006). Comparison between the modal identification method and the pod-galerkin method for model reduction in nonlinear diffusive systems. *Int. J. Numer. Meth. Engng*, 67(7) :895–915.

[Berdin et al., 2004] Berdin, C., Besson, J., Bugat, S., Desmorat, R., Feyel, F., Forest, S., Lorentz, E., Maire, E., Pardoen, T., Pineau, A., and Tanguy, B. (2004). *Local approach to fracture*. Presse de l'Ecole des Mines, France, Paris.

[Bergmann and Cordier, 2008] Bergmann, M. and Cordier, L. (2008). Optimal control of the cylinder wake in the laminar regime by trust-region methods and pod reduced-order models. *J. of Computational Physics*, 227 :7813–7840.

[Besson et al., 2001] Besson, J., Brocks, W., Chabanet, O., and Steglich, D. (2001). Ductile rupture of aluminum sheet materials. *European Journal of Finite Elements*, 10 :401–415.

[Biot, 1965] Biot, M. (1965). *Mechanics of Incremental Deformations*. Wiley, New York.

[Cordier and Bergmann, 2006] Cordier, L. and Bergmann, M. (2006). Réduction de dynamique par décomposition orthogonale aux valeurs propres (POD). Optimisation et contrôle des écoulements et des transferts, Ecole de printemps OCET, Aussois. 1-57.

[DeVore and Temlyakov, 1996] DeVore, R. and Temlyakov, V. (1996). Some remarks on greedy algorithms. *Adv. Comput. Math.*, 5 :173–187.

[Do, 2006] Do, T. (2006). *Optimisation de forme en forgeage 3D*. PhD thesis, Mines ParisTech.

[Fahl, 2000] Fahl, M. (2000). *Trust-region methods for flow control based on reduced order modeling*. PhD thesis, Trier University.

[Fritzen and Böhlke, 2010] Fritzen, F. and Böhlke, T. (2010). Three-dimensional finite element implementation of the nonuniform transformation field analysis. *Int. J. Numer. Meth. Engng*, 84 :803–829.

[Ganapathysubramanian and Zabaras, 2004] Ganapathysubramanian, S. and Zabaras, N. (2004). Design across length scales : a reduced-order model of polycrystal plasticity for the control of microstructure-sensitive material properties. *Comp. Meth. Appl. Mech. Engng.*, 193 :5017–5034.

[Germain, 1973] Germain, P. (1973). *Cours de mécanique des milieux continus*. Masson et Cie, Paris.

[Germain et al., 1983] Germain, P., Q.S., N., and Suquet, P. (1983). Continuum thermodynamics. *J. of Applied Mechanics*, 50 :1010–1020.

[Girault and Petit, 2005a] Girault, M. and Petit, D. (2005a). Identification methods in nonlinear heat conduction. part i : Model reduction. *International Journal of Heat and Mass Transfer*, 48 :105–118.

[Girault and Petit, 2005b] Girault, M. and Petit, D. (2005b). Identification methods in nonlinear heat conduction. part ii : inverse problem using a reduced model. *International Journal of Heat and Mass Transfer*, 48 :119–133.

[Gonzalez et al., 2010] Gonzalez, D., Ammar, A., Chinesta, F., and Cueto, E. (2010). Recent advances on the use of separated representations. *Int. J. Numer. Meth. Engng*, 81 :637–659.

[Halphen and Nguyen, 1975] Halphen, R. and Nguyen, Q. (1975). Sur les matériaux standards généralisés. *J. de Mec.*, 40 :39–63.

[Holmes et al., 1997] Holmes, P., Lumley, J., Berkooz, G., Mattingly, J., and Wittenberg, R. (1997). Low-dimensional models of coherents structures in turbulence. *Phys. Rep.*, 287 :337–384.

[Karhunen, 1946] Karhunen, K. (1946). Uber lineare methoden in der wahrscheinlichkeitsrechnung. *Annales Academiae Scientiarum Fennicae*, 37 :3–79.

[Ladevèze, 1985] Ladevèze, P. (1985). Sur une famille d'algorithmes en mécanique des structures. *C. R. Acad. Sci. Paris, Série II*, pages 41–44.

[Ladevèze and Nouy, 2003] Ladevèze, P. and Nouy, A. (2003). On a multiscale computational strategy with time and space homogenization for structural mechanics. *Comp. Meth. Appl. Mech. Engng.*, 192 :3061–3087.

[Lanczos, 1952] Lanczos, C. (1952). Solution of system of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand.*, 49 :33–53.

[Le Bris et al., 2009] Le Bris, C., Lelivre, T., and Maday, Y. (2009). Results and questions on a nonlinear approximation approach for solving high-dimensional partial differential equations. *Constructive Approximation*, 30 :621–651.

[Lemaitre and Chaboche, 1985] Lemaitre, J. and Chaboche, J. (1985). *Mécanique des matériaux solides*. first ed., Dunod, Paris(English version published by Cambridge University Press, Cambridge, 1990).

[Lieu et al., 2006] Lieu, T., Farhat, C., and Lesoinne, M. (2006). Reduced-order fluid/structure modeling of a complete aircraft configuration. *Computer Methods in Applied Mechanics and Engineering*, 195 :5730–5742.

[Loeve, 1963] Loeve, M. (1963). *Probability theory*. The University Series in Higher Mathematics. Van nostrand : Princeton, NJ.

[Lorentz et al., 2008] Lorentz, E., Besson, J., and Cano, V. (2008). Numerical simulation of ductile fracture : an efficient and robust implementation of the rousselier constitutive law. *Comp. Meth. Appl. Mech. Engrg.*, 197 :1965–1982.

[Lorenz, 1956] Lorenz, E. (1956). Empirical orthogonal functions and statistical weather predictions. Technical report, MIT, Departement of Meteorology, Scientific Report N1, Statistical Forecasting Project.

[Lumley, 1967] Lumley, J. (1967). *The structure of inhomogeneous turbulence*. Atmospheric Turbulence and Wave Propagation. A.M. Yaglom and V.I. Tatarski.

[Markovinovic and Jansen, 2006] Markovinovic, R. and Jansen, J. (2006). Accelerating iterative solution methods using reduced-order models as solution predictors. *Int. J. Numer. Meth. Engng*, 68 :525–541.

[Mazière et al., 2009] Mazière, M., Besson, J., Forest, S., Tanguy, B., Chalons, H., and Vogel, F. (2009). Overspeed burst of elastoviscoplastic rotating disks. part i : Analytical and numerical stability analyses. *Eur. J. Mech., A/Solids*, 28 :36–44.

[Michel and Suquet, 2003] Michel, J. and Suquet, P. (2003). Nonuniform transformation field analysis. *Int. J. Solids Structures*, 40 :6937–6955.

[Michel and Suquet, 2004] Michel, J. and Suquet, P. (2004). Computational analysis of nonlinear composite structures using the nonuniform transformation field analysis. *Comp. Meth. Appl. Mech. Engng.*, 193 :5477–5502.

[Monteiro et al., 2008] Monteiro, E., Yvonnet, J., and He, Q. (2008). Computational homogenization for nonlinear conduction in heterogeneous materials using model reduction. *Computational Materials Science*, 42(4) :704–712.

[Musienko et al., 2007] Musienko, A., Tatschl, A., Schmidegg, K., Kolednik, O., Pippan, R., and Cailletaud, G. (2007). Three-dimensional finite element simulation of a polycristalline copper specimen. *Acta Materiala*, 55 :4121–4136.

[Nelder and Mead, 1965] Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7(4) :308–313.

[Nguyen, 2007] Nguyen, N. (2007). A posteriori error estimation and basis adaptivity for reduced-basis approximation of nonaffine-parametrized linear elliptic partial differential equations. *J. of Computational Physics*, 227 :983–1006.

[Nicouleau et al., 2002] Nicouleau, E., Feyel, F., Quilicy, S., and G., C. (2002). Structural calculation and lifetime prediction in thermomechanical fatigue of engine components. *Eur. Struct. Integr. Soc.*, 29 :331–340.

[Niroormandi et al., 2008] Niroormandi, S., Alfaro, I., Cueto, E., and Chinesta, F. (2008). Real-time deformable models of non-linear tissues by model reduction techniques. *Comp. Meth. Prog. Biomed.*, 91.

[Ojalvo, 1962] Ojalvo, I. (1962). Conduction with time-dependent heat sources and boundary conditions : A modified separation-of-variables technique. *Int. J. Heat Mass Transfer.*, 5 :1105–1109.

[Park et al., 1999] Park, H., Chung, O., and Lee, J. (1999). On the solution of inverse heat transfer problem using the karhunen-loeve galerkin method. *International Journal of Heat and Mass Transfer*, 42 :127–142.

[Pelle and Ryckelynck, 2000] Pelle, J.-P. and Ryckelynck, D. (2000). An efficient adaptive strategy to master the global quality of viscoplastic analysis. *Computers and Structures*, 78.

[Pijaudier-Cabot and Benallal, 1993] Pijaudier-Cabot, G. and Benallal, A. (1993). Strain localization and bifurcation in a nonlocal continuum. *Int. J. Solids Structures*, 30 :1761–1775.

[Prulière et al., 2010] Prulière, E., Chinesta, F., and Ammar, A. (2010). On the deterministic solution of multidimensional parametric models using the proper generalized decomposition. *Math. and Comput. in Simulation*, 81(4) :791–810.

[Risler and Rey, 2000] Risler, F. and Rey, C. (2000). Iterative accelerating algorithms with krylov sub-spaces for the solution to large-scale non-linear problems. *Numer. Algorithms*, 23 :1–30.

[Rousselier, 1987] Rousselier, G. (1987). Ductile fracture models and their potential in local approach to fracture. *Nuclear Engineering and Design*, 105 :97–111.

[Ryckelynck, 2002] Ryckelynck, D. (2002). Réduction a priori de modèles thermomécaniques. *C. R. Mécanique*, 330 :499–505.

[Ryckelynck, 2005] Ryckelynck, D. (2005). A priori hyperreduction method : an adaptive approach. *Int. J. of Computational Physics*, 202 :346–366.

[Ryckelynck, 2009] Ryckelynck, D. (2009). Hyper reduction of mechanical models involving internal variables. *Int. J. Numer. Meth. Engng*, 77(1) :75–89.

[Ryckelynck, 2010] Ryckelynck, D. (2010). Toward "green" mechanical simulations in materials science : hyper-reduction of a polycrystal plasticity model. *Eur. J. Computational Mech.*, 19(4) :365–388.

[Ryckelynck et al., 2006] Ryckelynck, D., Chinesta, F., Cueto, E., and Amar, A. (2006). On the a priori model reduction : overview and recent developments. *Arch. Comput. Meth. Engng.*, 13(1) :91–128.

[Ryckelynck and Missoum-Benziane, 2010] Ryckelynck, D. and Missoum-Benziane, D. (2010). Multi-level a priori hyper reduction of mechanical

models involving internal variables. *Comp. Meth. Appl. Mech. Engng.*, 199 :1134–1142.

[Ryckelynck et al., 2011] Ryckelynck, D., Missoum-Benziane, D., Cartel, S., and Besson, J. (2011). A robust adaptive model reduction method for damage simulations. *Computational Materials Science*, 50 :1597–1605.

[Sarbandi et al., 2010] Sarbandi, B., Cartel, S., Besson, J., and Ryckelynck, D. (2010). Truncated integration for simultaneous simulation of sintering using a separated representation. *Arch. Comput. Meth. Engng.*, 1(1) :1–9.

[Schmidt, 1908] Schmidt, E. (1908). Zur theorie der linearen und nichtlinearen integralgleichungen. iii. teil. *Mathematische Annalen*, 65 :370–399.

[Sirovich, 1987] Sirovich, L. (1987). Turbulence and the dynamics of coherent structures parti : coherent structures. *Quaterly of applied mathematics*, (3) :561–571.

[Spalart et al., 1997] Spalart, P. R., Jou, W.-H., Strelets, M., and Allmaras, S. R. (1997). omments on the feasibility of les for wings, and on a hybrid rans/les approach. *Advances in DNS/LES, 1st AFOSR Int. Conf. On DNS/LES, Greyden Press, Columbus, OH,Aug 4-8.*

[Tanguy et al., 2005] Tanguy, B., Besson, J., Piques, R., and Pineau, A. (2005). Ductile to brittle transition of an a508 steel characterized by charpy impact test. part ii : Modeling of the charpy transition curve. *Engng Fracture Mechanics*, 72 :413–434.

[Temlyakov, 2000] Temlyakov, V. (2000). Weak greedy algorithms. *Adv. Comput. Math.*, 12 :213–227.

[Verdon et al., 2009] Verdon, N., Allery, C., Beghein, C., Hamdouni, A., and Ryckelynck, D. (2009). Reduced-order modelling for solving linear and non-linear equations. *Commun. Numer. Meth. Engrg.*, in press.

[Videcoq and Petit, 2001] Videcoq, E. and Petit, D. (2001). Model reduction for the resolution of multidimensional inverse heat conduction problems. *International Journal of Heat and Mass Transfer*, 44 :1899–1911.

[Xiao et al., 2009] Xiao, M., Breitkopf, P., Coelho, R., Knopf-Lenoir, C., Sidorkiewicz, M., and Villon, P. (2009). Model reduction by cpod and kriging. *Struct. Multidisc. Optim.*, 41 :555–574.

[Yvonnet and He, 2007] Yvonnet, J. and He, Q. (2007). The reduced model multiscale method (r3m) for the non-linear homogenization of hyperelastic media at finite strains. *J. of Computational Physics*, 223(1) :341–368.

[Ziegler, 1963] Ziegler, H. (1963). *Some extremum principles in irreversible thermodynamics with applications to continuum mechanics. In progress*

in Solid Mechanics, vol. IV. N.Sneddon, R. Hills (Eds), North-Holland, Amsterdam.

[Zienkiewicz and Taylor, 2000] Zienkiewicz, O. and Taylor, R. (2000). *Finite Element Method vols. 1-3*. Butterworth-Heinmann, London.

Annexe 1

```
#ifndef __REDUCED_STATE__
#define __REDUCED_STATE__

// ============================================================================
//   REDUCED_STATE   class for model reduction
//         q = A.a
//
//  03/19/07 related to APHR method D. Ryckelynck
// ============================================================================

#include <Global_client.h>
#include <DD_sub_domain.h>
#include <Array.h>
#include <Bool.h>
#include <File.h>
#include <Problem.h>
#include <Z_object.h>
#include <Zstream.h>
#include <List.h>

namespace ZSET {
  using namespace ZSET;

class MATRIX;
class VECTOR;
class REDUCED_ORDER_MODEL;


class REDUCED_STATE : public Z_OBJECT {
   public :
         REDUCED_ORDER_MODEL* its_ROM;
         DD_SUB_DOMAIN* My_SUB_DOMAIN;
         GLOBAL_CLIENT glc;

         int max_num_of_shape_fct;
         int Nb_Shape;
         int max_adap_without_selection;
         double epsilon_pod;
         double state_error;
         double norm_residu_before_iteration;
         double Significant_value;
         bool new_significant_value;
         double lambda_max_ref, lambda_max;
         double memory_factor;

         MATRIX A_buff;
         MATRIX a_buff;
         VECTOR a_incr;
         MATRIX b_buff;
         MATRIX Covariance_history_buff;
         SMATRIX GG;
         SMATRIX GG_tot;

         bool was_adapted;
         VECTOR Shared;

         // tihs class intend to support dof and internal variable related basis,
         // is_dof_basis bring support for parallel runing
```

```
51          // is_dof_basis = 1 , dof related basis ==> carful with global opératons
    over the hole structure ( shared stuff )
52          // is_dof_basis = 0 , Varint related basis ==> nothing special to do !!
53          bool is_dof_basis;
54          bool is_global_basis; // in case of parallel computation bases can be
55  global or related to each subdomain

56          int Inc_last_selection;
57          int Inc_last_update_cov;
58          int Size_last_selection;
59          int Gene_last_orthogonalization;
60          int Basis_generation;
61
62          REDUCED_STATE();
63          virtual ~REDUCED_STATE();

65  //        virtual void read_weight(RST_FSTREAM& obin);
66  //      virtual void subspace_adaptation(VECTOR& R, double Rsigni);         //
    expansion of ROM subpace and selection by POD
67          virtual void subspace_adaptation(VECTOR& R, double Rsigni, bool
    if_selection = TRUE);         // expansion of ROM subpace and selection by POD
68          virtual void subspace_adaptation_with_residue(VECTOR& R, double
    Rsigni);         // expansion of ROM subpace and selection by POD
69          virtual void subspace_selection();  // singular value decomposition for
    POD
70          virtual void subspace_expansion(VECTOR& R, double Rsigni, bool
    false_for_residue_true_for_state = TRUE);     // expansion of ROM subspace
71  //      virtual void subspace_expansion(VECTOR& R, double Rsigni);     // expansion
    of ROM subspace
72          virtual void subspace_resize(MATRIX& AA, int n_ex, int m_ex);          //
    resize the max_num_of_shape_fct
73          virtual void history_resize();
74          virtual void define_size(int n);
75          virtual void increase_buff();
76          virtual void manage_recurrent_event();  // management of memory of the ROM
77          virtual void init_rom_memory();
78          virtual void normalize_shape_fct();
79          virtual bool solve_var( VECTOR& Y, int Nb_selected, ARRAY<int>
    &iy_selected);
80          virtual bool solve_var( VECTOR& Y);
81          virtual bool solve_var( VECTOR& Y, VECTOR& ay);
82          virtual void update_significant_value();

84          virtual void read_rom(RST_FSTREAM& obin);
85          virtual void write_rom(RST_FSTREAM& rep);
86          virtual void write_rom_out();
87
88

89          virtual void initialize(REDUCED_ORDER_MODEL* its_PB);
90          virtual void polynomial_shape();
91          virtual void init_problem();
92          virtual void init_increment();
93          virtual void init_cov_hist();
94          virtual void update_cov_hist();
95          virtual void attenuate_cov_hist();
96          virtual LIST<int> sort(LIST<double>& beta);
97          virtual void power_method(DMATRIX &Lambda, MATRIX &Phi);
98          virtual void set_state(VECTOR& Y, double coef);  // for varint or varaux
```

```
 99          virtual void set_dof_increment(MESH* mesh); // for dof
100          virtual void set_dof_tot(MESH* mesh);
101
102          virtual void set_shape_fct(VECTOR& Y, int ishape);
103          virtual void set_shape_dof_tot(MESH* mesh, int ishape);
104
105          virtual void compute_GG();
106          virtual void compute_GG(int Nb_selected, ARRAY<int> &iy_selected);
107          virtual void ortho_shape_fct();
108          virtual void init_shape(int k);
109          virtual void give_global_A();
110
111
112          virtual void Eigenproblem(DMATRIX &Lambda, MATRIX &Phi);
113          virtual void tred2(int n, VECTOR &d, VECTOR &e, MATRIX &V );
114          virtual void tql2 (int n, VECTOR &d, VECTOR &e, MATRIX &V );
115
116  };
117  }
118  #endif
```

Annexe 2

```
#include <Array.h>
#include <Bool.h>
#include <Z_object.h>
#include <Zstream.h>
#include <Reduced_state.h>
#include <Utility.h>
#include <Global_matrix.h>
#include <Sparse_global_matrix.h>
#include <List.h>
#include <Calcul_timer.h>
#include <Reduced_order_model.h>
#include <Assert.h>
//
// HAcking around with flags to try and get NT to read a
// restart file
//
#ifdef _WIN32
  #define OUT_MODE ZIOS::out|ZIOS::binary|ZIOS::trunc
  #define IN_MODE ZIOS::in|ZIOS::binary
  #define Rflag "rb"
#else
  #define OUT_MODE ZIOS::out|ZIOS::trunc
  #define IN_MODE ZIOS::in
  #define Rflag "r"
#endif

  using namespace ZSET;


DECLARE_OBJECT(Z_OBJECT,REDUCED_STATE,APHR_STATE)


REDUCED_STATE::REDUCED_STATE()
{
   is_dof_basis = 0;
   is_global_basis = TRUE;
   max_num_of_shape_fct = 50;
   max_adap_without_selection = 3; // 20
   Nb_Shape = 0;
   epsilon_pod = 1.e-8;
   state_error = 1.e-3;

   Inc_last_selection = 0;
   Inc_last_update_cov = 0;
   Size_last_selection = Nb_Shape;

   Basis_generation = 0;
   Gene_last_orthogonalization = 5;

   Significant_value = 0.;
   new_significant_value = FALSE;
   lambda_max_ref = 0.;
   lambda_max = 0.;

   norm_residu_before_iteration = 0.;

   memory_factor = 1.;
```

```
      was_adapted = FALSE;
54  }

56  REDUCED_STATE::~REDUCED_STATE() {}

    /*
57  void REDUCED_STATE::subspace_adaptation(VECTOR& R, double Rsigni) // purely local
58  {
59      subspace_adaptation( R, Rsigni, TRUE); // purely local
60  }
62  */

63  //void  REDUCED_STATE::read_weight(RST_FSTREAM& obin) {}


64  void REDUCED_STATE::subspace_adaptation(VECTOR& R, double Rsigni, bool
    if_selection) // purely local
65  {
66     Timer_counter.start_time("REDUCED_STATE::Subspace_adaptation");
67     subspace_expansion(R, Rsigni);
68
69    if ((Nb_Shape > Zmax(0,Size_last_selection+max_adap_without_selection) ) &
70  (its_ROM->No_selection == FALSE)) {

        if (if_selection) subspace_selection();
72    }

      if (Basis_generation == 25*(Basis_generation/25)) compute_GG();
73    Timer_counter.stop_time("REDUCED_STATE::Subspace_adaptation");
75  }

    void REDUCED_STATE::subspace_adaptation_with_residue(VECTOR& R, double Rsigni)
76  {
77     Timer_counter.start_time("REDUCED_STATE::Subspace_adaptation_with_residue");
78     bool false_for_residue_true_for_state = FALSE;
79
80  //     orthogonalisation
81    VECTOR a_R(Nb_Shape);
82    bool isok_dof;
84    isok_dof = solve_var( R, a_R );  // optimization of a_R

      if (Nb_Shape > Zmax(0,Size_last_selection+max_adap_without_selection) )  {
85       subspace_selection();
87    }

      for (int i=0;i<R.size();i++) {
88       for (int j=0;j<Nb_Shape;j++)  R[i] -= A_buff(i,j)*a_R[j];
90    }

      subspace_expansion(R, Rsigni, false_for_residue_true_for_state);
91    Timer_counter.stop_time("REDUCED_STATE::Subspace_adaptation_with_residue");
93  }

    void REDUCED_STATE::subspace_selection() // purely local
94  {
95    Timer_counter.start_time("REDUCED_STATE::Subspace_selection");
96    if (Nb_Shape > 0) {
97       int Nb_increments = its_ROM->Nb_increments;
```

```
98        int i_max = 0;
99
100       MATRIX U;
101       DMATRIX W;
102
103       ARRAY<int> sel(Nb_Shape);
104
105       update_cov_hist();
106
107     //power_method(W, U);
108       Eigenproblem(W, U);
109
          lambda_max = lambda_max_ref;
110       i_max = 0;
111       for (int i=0;i<W.n;i++) {
112         if (fabs(W(i,i)) > lambda_max) {
113           lambda_max = fabs(W(i,i));
114           i_max = i;
115         }
116       }
117
          int new_Nb_shape = 0;
118       for (int i=0;i<W.n;i++)  {
119        if (fabs(W(i,i)) <= epsilon_pod * lambda_max) sel[i] = 0;
120        else {
121             sel[i] = 1;
122             new_Nb_shape += 1;
123          }
124        }
125       if  (new_Nb_shape == 0) {
126        new_Nb_shape = 1;
127        sel[0]=1;
128       }
129
130       MATRIX AA(A_buff.n,Nb_Shape);
131       MATRIX a(Nb_Shape,Nb_increments);
132       MATRIX cov(Nb_Shape,Nb_Shape);
133
134       for (int it=0;it<Nb_increments;it++)  {
135        for (int j=0; j<Nb_Shape; j++)  {
136            a(j,it) = a_buff(j,it);
137            a_buff(j,it) = 0.;
138            }
139       }
140
141       for (int i=0; i<new_Nb_shape; i++) {
142        if (sel[i] == 1)  {
143          for (int it=0;it<Nb_increments;it++)  {
144             for (int j=0; j<Nb_Shape; j++)  a_buff(i,it) += U(j,i)*a(j,it);
145          }
146        }
147       }
148
          for (int id=0;id<A_buff.n;id++)  {
149        for (int j=0; j<Nb_Shape; j++)  {
150            AA(id,j) = A_buff(id,j);
151            A_buff(id,j) = 0.;
152            }
```

```
152         }
154
            for (int i=0; i<new_Nb_shape; i++) {
155            if (sel[i] == 1)  {
156              for (int id=0;id<A_buff.n;id++)  {
157               for (int j=0; j<Nb_Shape; j++)  {
158                   A_buff(id,i) += AA(id,j)*U(j,i);
159               }
160              }
161            }
162         }
163
            for (int it=0;it<Nb_Shape;it++)  {
164            for (int j=0; j<Nb_Shape; j++)  {
165                cov(j,it) = Covariance_history_buff(j,it);
166                Covariance_history_buff(j,it) = 0.;
167            }
168         }
169         MATRIX PROD(Nb_Shape,new_Nb_shape);
170
171         for (int it=0;it<new_Nb_shape;it++)  {
172            for (int j=0; j<Nb_Shape; j++)  PROD(j,it) = 0.;
173         }
174
175         for (int i=0; i<new_Nb_shape; i++) {
176            if (sel[i] == 1)  {
177              for (int it=0;it<Nb_Shape;it++)  {
178                 for (int j=0; j<Nb_Shape; j++)  PROD(it,i) += cov(it,j)*U(j,i);
179              }
180            }
181         }
182
            for (int i=0; i<new_Nb_shape; i++) {
183            if (sel[i] == 1)  {
184              for (int it=0;it<Nb_Shape;it++)  {
185                 for (int j=0; j<new_Nb_shape; j++)  Covariance_history_buff(i,j) += U
    (it,i)*PROD(it,j) ;
186              }
187            }
188         }
189
190         MATRIX cov_check(new_Nb_shape,new_Nb_shape,Covariance_history_buff,0,0);
191         if ( norm(cov_check-transpose(cov_check)) > 1.e-6*norm(cov_check) ) {
192             Out<<cov(0,0)<<endl;
193            for (int i=0;i<cov_check.n;i++) {
194             for (int j = 0;j<cov_check.m;j++) Out<<cov_check(i,j);
195             Out<<endl;
196            }
197            ERROR(" the matrix must be symetric");
198         }
199
200
201         if (its_ROM->extrapolate_varint) {

              Timer_counter.start_time("compute_GG");
202           for (int it=0;it<new_Nb_shape;it++)  {
203              for (int j=0; j<Nb_Shape; j++)  PROD(j,it) = 0.;
204           }
205
```

```
          for (int i=0; i<new_Nb_shape; i++) {
206          if (sel[i] == 1)  {
207            for (int it=0;it<Nb_Shape;it++)  {
208               for (int j=0; j<Nb_Shape; j++)  PROD(it,i) += GG_tot(it,j)*U(j,i);
209            }
210          }
211       }

          GG_tot.resize(new_Nb_shape);
213       for (int i=0; i<new_Nb_shape; i++) {
214          if (sel[i] == 1)  {
215             for (int j=0; j<i+1; j++) {
216               GG_tot(i,j) = 0.;
217               for (int it=0;it<Nb_Shape;it++)  GG_tot(i,j) += U(it,i)*PROD(it,j) ;
218                  GG_tot(j,i) = GG_tot(i,j);
219             }
220          }
221       }
222       Timer_counter.stop_time("compute_GG");
224     }

        Nb_Shape = new_Nb_shape;
225   }
226   Size_last_selection = Nb_Shape;
227   Timer_counter.stop_time("REDUCED_STATE::Subspace_selection");
229 }

    /*
230 void REDUCED_STATE::subspace_expansion(VECTOR& R, double Rsigni)
231 {
232   subspace_expansion( R, Rsigni, TRUE);
233 }
235 */

    void REDUCED_STATE::subspace_expansion(VECTOR& R, double Rsigni, bool
    false_for_residue_true_for_state) // parallele ready
236 {
237    Timer_counter.start_time("REDUCED_STATE::Subspace_expansion");
238
239    VECTOR GLC_temp_g_sum(1);
240    VECTOR R_temp(R.size());
241    double norme_R ;
242    if(its_ROM->reference_problem->Is_parallel_computation && is_global_basis) {
243       if (is_dof_basis)   {
244          for (int i=0; i<R.size(); i++) R_temp[i] = R[i]/ sqrt(Shared[i]) ;
245          GLC_temp_g_sum[0] = norm(R_temp);
246       } else {
247          GLC_temp_g_sum[0] = norm(R);
248       }
249       GLC_temp_g_sum[0]  *= GLC_temp_g_sum[0];
250       glc.g_sum(GLC_temp_g_sum);
251       GLC_temp_g_sum[0] = sqrt(GLC_temp_g_sum[0]);
252       norme_R = GLC_temp_g_sum[0];
253    }
255    else norme_R = norm(R);

       int  Nb_increments = its_ROM->Nb_increments;
256
```

```
//  if (norme_R > state_error*Rsigni + 0.1 * its_ROM->rom_error * Significant_value*sqrt(A_buff.n)) {  // DR 14Aug2008 introduction of state_error
  if (norme_R > state_error/10.*Rsigni + state_error/100. * Significant_value*sqrt(A_buff.n)) {  // DR 14Aug2008 introduction of state_error

     Basis_generation +=1;
     was_adapted = TRUE;
     int i;
     increase_buff();
     Nb_Shape += 1;
     MATRIX AA(A_buff.n,Nb_Shape,A_buff,0,0);
     MATRIX a(Nb_Shape,Nb_increments,a_buff,0,0);
     MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);

     for (i=0;i<Nb_increments;i++) a(Nb_Shape-1,i) = 0.;
     if ((false_for_residue_true_for_state) & (Nb_increments > 0)) a(Nb_Shape-1,Nb_increments-1) = norme_R;

     for (i=0;i<AA.n;i++) AA(i,Nb_Shape-1) = R[i]/norme_R;

     for (i=0;i<Nb_Shape;i++) {
         cov(Nb_Shape-1,i) = 0.;
         cov(i,Nb_Shape-1) = 0.;
     }

     double GG_tot_contrib;
     VECTOR GG_tot_col(Nb_Shape);

     if (its_ROM->extrapolate_varint) {
      if (GG_tot.n > Nb_Shape) {
        compute_GG();
      }
      else {
        Timer_counter.start_time("compute_GG");
        subspace_resize(GG_tot, Nb_Shape, Nb_Shape);
///%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%///
//      si c'est relatif au Varint ==> somme directe des contribution par sous domaine
//      si c'est relatif au Dof    ==> somme avec prise en compte du recouvrement aux interfaces
///%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%///

        for (int i=0; i<Nb_Shape; i++) {
           GG_tot_col[i] = 0.;
           for (int j=0; j<AA.n; j++){
              GG_tot_contrib = AA(j,i)*AA(j,Nb_Shape-1);
              if (its_ROM->reference_problem->Is_parallel_computation && is_dof_basis && is_global_basis)  GG_tot_contrib /= Shared[j];
              GG_tot_col[i] += GG_tot_contrib;
           }
        }
        if (its_ROM->reference_problem->Is_parallel_computation && GG_tot.size()>0 && is_global_basis)  glc.g_sum(GG_tot_col);

        for (int i=0; i<Nb_Shape; i++) {
           GG_tot(i,Nb_Shape-1) = GG_tot_col[i];
```

```
302             GG_tot(Nb_Shape-1,i) = GG_tot(i,Nb_Shape-1);
303           }
304           Timer_counter.stop_time("compute_GG");
305         }
306       }
307     }
308     else {
309       was_adapted = FALSE;
310 //    Out<<"No expansion : | R_state | = "<<norme_R/(state_error*Rsigni +
    Significant_value)<<endl;
311     }
312     Timer_counter.stop_time("REDUCED_STATE::Subspace_expansion");
313 }
314
    void REDUCED_STATE::increase_buff()
315 {
316    Timer_counter.start_time("REDUCED_STATE::Increase_buff");
317    int const Np = 20;
318      if (Nb_Shape+1 > A_buff.m) {
319         subspace_resize(A_buff, A_buff.n, Nb_Shape+Np);
320         subspace_resize(a_buff, Nb_Shape+Np, a_buff.m);
321         subspace_resize(b_buff, Nb_Shape+Np, b_buff.m);
322         subspace_resize(Covariance_history_buff, Nb_Shape+Np, Nb_Shape+Np);
323         a_incr.resize(Nb_Shape+Np);
324         max_num_of_shape_fct = Nb_Shape+Np;
325      }
326      Timer_counter.stop_time("REDUCED_STATE::Increase_buff");
327
328 }
    void REDUCED_STATE::subspace_resize(MATRIX& AA, int n_ex, int m_ex )  // purely
    local
329 {
330    Timer_counter.start_time("REDUCED_STATE::Subspace_resize");
331    if (n_ex < AA.n) ERROR("n_ex < AA.n");
332    if (m_ex < AA.m) ERROR("n_ex < AA.m");
333
334    if ((AA.n == 0) || (AA.m == 0)){
335       AA.resize(n_ex, m_ex);
336       AA = 0.;
337    }else{
338       MATRIX BB;
339       BB = AA;
340       AA.resize(n_ex, m_ex);
341       AA = 0.;
342       MATRIX CC(BB.n,BB.m,AA,0,0);
343       CC = BB;
344    }
345    Timer_counter.stop_time("REDUCED_STATE::Subspace_resize");
346
347 }
    void REDUCED_STATE::history_resize()  // purely local
348 {
349    subspace_resize(a_buff, max_num_of_shape_fct, its_ROM->max_num_of_time_steps);
350
351 }
    void REDUCED_STATE::define_size(int n)  // purely local
352 {
353    Timer_counter.start_time("REDUCED_STATE::Define_size");
```

```
   int max_num_of_time_steps = its_ROM->max_num_of_time_steps;
   A_buff.resize(n,max_num_of_shape_fct);
   a_buff.resize(max_num_of_shape_fct,max_num_of_time_steps);
   a_incr.resize(max_num_of_shape_fct);
   b_buff.resize(max_num_of_shape_fct,max_num_of_time_steps);
   Covariance_history_buff.resize(max_num_of_shape_fct,max_num_of_shape_fct);
   Timer_counter.stop_time("REDUCED_STATE::Define_size");
}

void REDUCED_STATE::manage_recurrent_event() // purely local
{
   Timer_counter.start_time("REDUCED_STATE::Manage_recurrent_event");
   if (Nb_Shape > 0) {
     MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);

     cov *= its_ROM->gamma_history;
   }
  Timer_counter.stop_time("REDUCED_STATE::Manage_recurrent_event");
}

void REDUCED_STATE::init_rom_memory() // purely local
{
   Timer_counter.start_time("REDUCED_STATE::Init_rom_memory");
   if (Nb_Shape > 0) {
     MATRIX a_dof(Nb_Shape,its_ROM->Nb_increments,a_buff,0,0);
     MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);

     cov = a_dof*transpose(a_dof);

   }
  Timer_counter.stop_time("REDUCED_STATE::Init_rom_memory");
}

void REDUCED_STATE::normalize_shape_fct() // parallele ready // a voir
{
   Timer_counter.start_time("REDUCED_STATE::Normalize_shape_fct");
   DMATRIX W(Nb_Shape);
  double norm_shape;

  for (int i=0; i< Nb_Shape; i++)  {
     MATRIX ColA(A_buff.n,1,A_buff,0,i);
     VECTOR temp_ColA(A_buff.n); for(int j=0;j<A_buff.n;j++) temp_ColA[j] = ColA(j,0);

     MATRIX Liga(1,a_buff.m,a_buff,i,0);
     MATRIX Ligcov(1,Nb_Shape,Covariance_history_buff,i,0);
     MATRIX Colcov(Nb_Shape,1,Covariance_history_buff,0,i);


///%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%///
// parallélisation du calcul de norme
///%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%///
   VECTOR GLC_temp_g_sum(1);
   if(its_ROM->reference_problem->Is_parallel_computation && is_global_basis) {
      norm_shape = its_ROM->parallel_norm(temp_ColA, is_dof_basis);
   } else norm_shape = norm(temp_ColA);
```

```
      ColA /= norm_shape;
      Liga *= norm_shape;
      Ligcov *= norm_shape;
      Colcov *= norm_shape;
   }
   Timer_counter.stop_time("REDUCED_STATE::Normalize_shape_fct");
}

void REDUCED_STATE::compute_GG() // parallele ready
{
   Timer_counter.start_time("REDUCED_STATE::Compute_GG_tot");

   GG_tot.resize(Nb_Shape);
///%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%///
//       si c'est relatif au Varint ==> somme directe des contribution par sous domaine
//       si c'est relatif au Dof    ==> somme avec prise en compte du recouvrement aux interfaces
///%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%///

   if (its_ROM->reference_problem->Is_parallel_computation && is_dof_basis && is_global_basis)
   {
      for (int ishape=0;ishape<Nb_Shape;ishape++)  {
         for (int jshape=0;jshape<ishape;jshape++)  {
            GG_tot(ishape,jshape) = 0.;
            for (int i=0;i<A_buff.n;i++){
               GG_tot(ishape,jshape) += (A_buff(i,ishape) * A_buff(i,jshape))/Shared[i];
            }
            GG_tot(jshape,ishape) = GG_tot(ishape,jshape);
         }
         GG_tot(ishape,ishape) = 0.;
         for (int i=0;i<A_buff.n;i++){
            GG_tot(ishape,ishape) += (A_buff(i,ishape) * A_buff(i,ishape))/Shared[i];
         }
      }
   }
   else {
      for (int ishape=0;ishape<Nb_Shape;ishape++)  {
         for (int jshape=0;jshape<ishape;jshape++)  {
            GG_tot(ishape,jshape) = 0.;
            for (int i=0;i<A_buff.n;i++){
               GG_tot(ishape,jshape) += A_buff(i,ishape) * A_buff(i,jshape);
            }
            GG_tot(jshape,ishape) = GG_tot(ishape,jshape);
         }
         GG_tot(ishape,ishape) = 0.;
         for (int i=0;i<A_buff.n;i++){
            GG_tot(ishape,ishape) += A_buff(i,ishape) * A_buff(i,ishape);
         }
      }
   }
```

```
452
453      if (its_ROM->reference_problem->Is_parallel_computation && GG_tot.size()>0 &&
     is_global_basis)  glc.g_sum(GG_tot);
454      Timer_counter.stop_time("REDUCED_STATE::Compute_GG_tot");
455
456   }

      void REDUCED_STATE::compute_GG(int Nb_selected, ARRAY<int> &iy_selected)  //
      parallele ready
457   {
458      Timer_counter.start_time("REDUCED_STATE::Compute_GG");
459
460      if (its_ROM->hyper_red_request)  {
461         int iy;
462         GG.resize(Nb_Shape);
463         if (its_ROM->reference_problem->Is_parallel_computation && is_dof_basis &&
     is_global_basis)
464         {
465            for (int i=0;i<Nb_Shape;i++) {
466               for (int j=0;j<i;j++) {
467                  GG(i,j) = 0.;
468                  for (int ii=0;ii<Nb_selected;ii++) {
469                     iy = iy_selected[ii];
470                     GG(i,j) += (A_buff(iy,i)*A_buff(iy,j))/ Shared[iy];
471                  }
472                  GG(j,i) = GG(i,j);
473               }
474               GG(i,i) = 0.;
475               for (int ii=0;ii<Nb_selected;ii++) {
476                  iy = iy_selected[ii];
477                  GG(i,i) += (A_buff(iy,i)*A_buff(iy,i))/ Shared[iy];
478               }
479            }
480         }
481         else{
482            for (int i=0;i<Nb_Shape;i++) {
483               for (int j=0;j<i;j++) {
484                  GG(i,j) = 0.;
485                  for (int ii=0;ii<Nb_selected;ii++) {
486                     GG(i,j) += A_buff(iy_selected[ii],i)*A_buff(iy_selected[ii],j);
487                  }
488                  GG(j,i) = GG(i,j);
489               }
490               GG(i,i) = 0.;
491               for (int ii=0;ii<Nb_selected;ii++) {
492                  GG(i,i) += A_buff(iy_selected[ii],i)*A_buff(iy_selected[ii],i);
493               }
494            }
495         }
496      }
497      if (its_ROM->reference_problem->Is_parallel_computation && GG.size()>0 &&
     is_global_basis)  glc.g_sum(GG);
498      Timer_counter.stop_time("REDUCED_STATE::Compute_GG");
499
500
501   }

      bool REDUCED_STATE::solve_var( VECTOR& Y, int Nb_selected, ARRAY<int>
      &iy_selected)    // parallele ready
502   {
```

```
503      Timer_counter.start_time("REDUCED_STATE::Solve_var");
504      int  Nb_increments = its_ROM->Nb_increments;
505    bool isok = TRUE;
506    if (Nb_Shape > 0)  {
507      MATRIX a(Nb_Shape,1,a_buff,0,Nb_increments-1);
508      VECTOR F_rom(Nb_Shape);
509      VECTOR ay(Nb_Shape);
510      VECTOR Y_estimate(A_buff.n);
511
         int iy;
512      double F_rom_contrib;
513
514      if (its_ROM->true_for_hyper_reduction == FALSE) {
515         isok = solve_var(Y);
516      }
517      else {
518          for (int i=0;i<Nb_Shape;i++) {
519                F_rom[i] = 0.;
520                for (int ii=0;ii<Nb_selected;ii++) {
521                   iy = iy_selected[ii];
522                   F_rom_contrib = A_buff(iy,i)*Y[iy];
523                   if (its_ROM->reference_problem->Is_parallel_computation &&
     is_dof_basis && is_global_basis)  F_rom_contrib /= Shared[iy];
524                   F_rom[i] += F_rom_contrib;
525                }
526          }
527          if (its_ROM->reference_problem->Is_parallel_computation && F_rom.size()>0
     && is_global_basis)  glc.g_sum(F_rom);
528          isok = GG.gauss_solver(F_rom, ay, TRUE);
529          if (!isok) {
530              Out<<"GG is modified"<<endl;
531              for (int i=1;i<Nb_Shape;i++) GG(i,i) += i*1.e-8*GG(0,0);
532              isok = GG.gauss_solver(F_rom, ay, TRUE);
533
534              if (!isok) return isok;
535          }
536          for (int i=0;i<Nb_Shape;i++) a(i,0) = ay[i];
537      }

       }
538    Timer_counter.stop_time("REDUCED_STATE::Solve_var");
539    return isok;
540  }
541
     bool REDUCED_STATE::solve_var( VECTOR& Y)   // parallele ready
542  {
543     Timer_counter.start_time("REDUCED_STATE::Solve_var");
544     int  Nb_increments = its_ROM->Nb_increments;
545    bool isok = TRUE;
546
547    if (Nb_Shape > 0)  {
548      MATRIX a(Nb_Shape,1,a_buff,0,Nb_increments-1);
549
550      VECTOR F_rom(Nb_Shape);
551      MATRIX R_rom(Nb_Shape,1);
552      VECTOR ay(Nb_Shape);
553      MATRIX Y_estimate(A_buff.n,1);
554      double F_rom_contrib;
```

```
555
          for (int i=0;i<Nb_Shape;i++) {
556          F_rom[i] = 0.;
557          for (int iy=0;iy<A_buff.n;iy++) {
558             F_rom_contrib = A_buff(iy,i)*Y[iy];
559             if (its_ROM->reference_problem->Is_parallel_computation && is_dof_basis
      && is_global_basis)  F_rom_contrib /= Shared[iy];
560             F_rom[i] += F_rom_contrib;
561          }
562       }
563       if (its_ROM->reference_problem->Is_parallel_computation && F_rom.size()>0 &&
      is_global_basis)  glc.g_sum(F_rom);
564
565       isok = GG_tot.gauss_solver(F_rom, ay, TRUE);
566       if (!isok) return isok;
567
568       for (int i=0;i<Nb_Shape;i++) a(i,0) = ay[i];
569
570     }
571     Timer_counter.stop_time("REDUCED_STATE::Solve_var");
572     return isok;
573   }
574
      bool REDUCED_STATE::solve_var( VECTOR& Y, VECTOR& ay)   // parallele ready
575   {
576      Timer_counter.start_time("REDUCED_STATE::Solve_var");
577      int  Nb_increments = its_ROM->Nb_increments;
578     bool isok = TRUE;
579     if (Nb_Shape > 0)  {
580       MATRIX a(Nb_Shape,1,a_buff,0,Nb_increments-1);
581
          VECTOR F_rom(Nb_Shape);
582       MATRIX R_rom(Nb_Shape,1);
583       MATRIX Y_estimate(A_buff.n,1);
584       double F_rom_contrib;
585
          for (int i=0;i<Nb_Shape;i++) {
586          F_rom[i] = 0.;
587          for (int iy=0;iy<A_buff.n;iy++) {
588             F_rom_contrib = A_buff(iy,i)*Y[iy];
589             if (its_ROM->reference_problem->Is_parallel_computation && is_dof_basis
      && is_global_basis)  F_rom_contrib /= Shared[iy];
590             F_rom[i] += F_rom_contrib;
591          }
592       }
593       if (its_ROM->reference_problem->Is_parallel_computation && F_rom.size()>0 &&
      is_global_basis)  glc.g_sum(F_rom);
594
595       isok = GG_tot.gauss_solver(F_rom, ay, TRUE);
596       if (!isok) return isok;
597
598     }
599     Timer_counter.stop_time("REDUCED_STATE::Solve_var");
600     return isok;
601   }
602


603   void REDUCED_STATE::update_significant_value()
```

```
604  {
605     int  Nb_increments = its_ROM->Nb_increments;
606
607     if (Nb_Shape > 0) {
608       double maximum = 0.;
609           for (int i=0;i<A_buff.n;i++) maximum = Zmax(maximum, fabs(A_buff(i,0)));
610           Significant_value = maximum;
611           maximum = 0.;
612           for (int i=0;i<Nb_increments;i++) maximum = Zmax(maximum, fabs(a_buff
     (0,i)));
613           Significant_value *= sqrt(epsilon_pod) * maximum;
614     }
615  }
616
617  void REDUCED_STATE::read_rom(RST_FSTREAM& file)  // not yet taged
618  {
619     Timer_counter.start_time("REDUCED_STATE::Read_rom");
620     int  max_num_of_time_steps = its_ROM->max_num_of_time_steps;
621    int  Nb_increments = its_ROM->Nb_increments;
622    int nb_block;
623    double Significant_value_old;
624
625    if (its_ROM->multidimensional_basis_extension) nb_block = GPINT
     ("Solver.DBSM_Nb_Block");
626    else nb_block = 1;
627
       file.read( &Nb_Shape, sizeof(int));
628    file.read( &Size_last_selection, sizeof(int));
629    file.read( &Inc_last_selection, sizeof(int));
630    file.read( &Inc_last_update_cov, sizeof(int));
631    file.read( &Basis_generation, sizeof(int));
632    file.read( &Gene_last_orthogonalization, sizeof(int));
633    file.read( &Significant_value_old, sizeof(double));
634    if (~new_significant_value) Significant_value = Significant_value_old;
635
636    if (a_buff.m < max_num_of_time_steps)  a_buff.resize
     (max_num_of_shape_fct,max_num_of_time_steps);
637
638    if (Nb_Shape > max_num_of_shape_fct) {
639       max_num_of_shape_fct = Nb_Shape + 50;
640       A_buff.resize(A_buff.n,max_num_of_shape_fct);
641       a_buff.resize(max_num_of_shape_fct,max_num_of_time_steps);
642       Covariance_history_buff.resize(max_num_of_shape_fct,max_num_of_shape_fct);
643    }
644
645  double somme  = 0;
646
647    // loop on dof
648    for (int i=0;i<A_buff.n/nb_block;i++) {
649  //  loop on shape functions
650         for (int kshape=0;kshape<Nb_Shape;kshape++) {
651             file.read(DBL_CAST &A_buff(i,kshape), sizeof(double));
652             somme += A_buff(i,kshape);
653         }
```

```
654     }
655   // in case of multidimensional analysis using a rom from a single simulation
656     for (int j=1;j<nb_block;j++) {
657        int i;
658        i = A_buff.n/nb_block;
659        for (int j=0;j<A_buff.n/nb_block;j++) {
660         for (int kshape=0;kshape<Nb_Shape;kshape++) A_buff(i,kshape) = A_buff
      (j,kshape);
661         i++;
662        }
663     }


665   somme  = 0;
666   // time functions
667   // loop on increments
668     for (int i=0;i<Nb_increments;i++) {
669   //  loop on shape functions
670         for (int kshape=0;kshape<Nb_Shape;kshape++) {
671             file.read(DBL_CAST &a_buff(kshape,i), sizeof(double));
672             somme += a_buff(kshape,i);
673         }
674     }

      somme  = 0;
676   // Covariance
677   // loop on ishape
678     for (int ishape=0;ishape<Nb_Shape;ishape++) {
679   //  loop on shape functions
680         file.read(DBL_CAST &Covariance_history_buff(ishape,ishape), sizeof(double));
681         for (int kshape=ishape+1;kshape<Nb_Shape;kshape++) {
682             file.read(DBL_CAST &Covariance_history_buff(ishape,kshape), sizeof
      (double));
683             Covariance_history_buff(kshape,ishape) = Covariance_history_buff
      (ishape,kshape);
684             somme += Covariance_history_buff(ishape,kshape);
685         }
686     }
687     Timer_counter.stop_time("REDUCED_STATE::Read_rom");
688   }

      void REDUCED_STATE::write_rom(RST_FSTREAM& file) // not yet taged
690   {
691      Timer_counter.start_time("REDUCED_STATE::Write_rom");
692      int  Nb_increments = its_ROM->Nb_increments;
693
694     file.write( &Nb_Shape, sizeof(int));
695     file.write( &Size_last_selection, sizeof(int));
696     file.write( &Inc_last_selection, sizeof(int));
697     file.write( &Inc_last_update_cov, sizeof(int));
698     file.write( &Basis_generation, sizeof(int));
699     file.write( &Gene_last_orthogonalization, sizeof(int));
700     file.write( &Significant_value, sizeof(double));
701
702     // loop on dof
703     for (int i=0;i<A_buff.n;i++) {
704   //  loop on shape functions
705         for (int kshape=0;kshape<Nb_Shape;kshape++) {
```

```
706             file.write(DBL_CAST &A_buff(i,kshape), sizeof(double));
707         }
708     }

    // time functions
710 // loop on increments
711   for (int i=0;i<Nb_increments;i++) {
712 //  loop on shape functions
713       for (int kshape=0;kshape<Nb_Shape;kshape++) {
714           file.write(DBL_CAST &a_buff(kshape,i), sizeof(double));
715       }
716     }

    // Covariance
718 // loop on ishape
719   for (int ishape=0;ishape<Nb_Shape;ishape++) {
720 //  loop on shape functions
721       file.write(DBL_CAST &Covariance_history_buff(ishape,ishape), sizeof(double));
722       for (int kshape=ishape+1;kshape<Nb_Shape;kshape++) {
723           file.write(DBL_CAST &Covariance_history_buff(ishape,kshape), sizeof
    (double));
724       }
725     }

727   Timer_counter.stop_time("REDUCED_STATE::Write_rom");


728


729


730


    //Modif SC 10-06-16
731 //Calcul des mu_ij pour le calcul de l'intensite des ER (en post_traitement)


733   if (is_dof_basis){
734     MATRIX mata(Nb_Shape,Nb_increments,a_buff,0,0);
735     VECTOR mu_ij(Nb_Shape);
736     double sommeincr;
737     for (int kshape=0;kshape<Nb_Shape;kshape++) {
738     sommeincr=0.;
739       for (int si=0;si<Nb_increments;si++) {
740         sommeincr=sommeincr+pow(mata(kshape,si),2);
741       }
742       mu_ij[kshape]=sommeincr;
743     }

      STRING file_name;
745     file_name = "mu_ij.dat";
746     Zofstream  Kk_file(file_name(),ios::app);
747     Kk_file.setf(ios::scientific,ios::floatfield);
748     Kk_file.precision(8);
749     Kk_file <<mu_ij.size()<<"\t";
750     for (int i=1;i<Nb_Shape+1;i++){
751       Kk_file <<mu_ij[i]<<"\t";
752     }
753      Kk_file <<"\n";
```

```
754       Kk_file.flush();
755       Kk_file.close();
756     }
757     //Fin Modif SC 10-06-16
758
759
760
761
762
763
764   }

      void REDUCED_STATE::write_rom_out() // not yet taged
765   {
766      Timer_counter.start_time("REDUCED_STATE::Write_rom_out");
767      int  Nb_increments = its_ROM->Nb_increments;
768
        Out<<"Nb_increments"<<Nb_increments<<endl;
769
770     Out<<"Nb_Shape"<<Nb_Shape<<endl;
771     Out<<"Size_last_selection"<<Size_last_selection<<endl;
772     Out<<"Inc_last_selection"<<Inc_last_selection<<endl;
773   /*
774     // loop on dof
775     for (int i=0;i<A_buff.n;i++) {
776   //  loop on shape functions
777         for (int kshape=0;kshape<Nb_Shape;kshape++) {
778             Out<<A_buff(i,kshape);
779         }
780         Out<<endl;
781     }
782
      // time functions
783   // loop on increments
784     for (int i=0;i<Nb_increments;i++) {
785   //  loop on shape functions
786         for (int kshape=0;kshape<Nb_Shape;kshape++) {
787             Out<<a_buff(kshape,i);
788         }
789         Out<<endl;
790     }
791
      // Covariance
792   // loop on ishape
793     for (int ishape=0;ishape<Nb_Shape;ishape++) {
794   //  loop on shape functions
795         Out<<Covariance_history_buff(ishape,ishape);
796         for (int kshape=ishape+1;kshape<Nb_Shape;kshape++) {
797             Out<<Covariance_history_buff(ishape,kshape);
798         }
799         Out<<endl;
800     }
801   */
```

```
802     Timer_counter.stop_time("REDUCED_STATE::Write_rom_out");
803   }
804
805   void REDUCED_STATE::initialize(REDUCED_ORDER_MODEL* its_PB) // parallele ready
806   {
807      Timer_counter.start_time("REDUCED_STATE::Initialize");
808      its_ROM = its_PB;
809      My_SUB_DOMAIN = its_ROM->reference_problem->mesh->get_domain();
810
811      if (its_ROM->reference_problem->Is_parallel_computation){
812         Shared.resize(My_SUB_DOMAIN->nb_dd_dofs());
813         Shared = 1. ;
814         for (int i=0; i<Shared.size();i++) {
815           if (My_SUB_DOMAIN->give_dd_dof(i))
816             Shared[My_SUB_DOMAIN->give_dd_dof(i)->dof->rank()] = My_SUB_DOMAIN-
    >give_dd_dof(i)->shared;
817         }
818      }
819   /*
820   permet l'acces aux fonctions :
821   My_SUB_DOMAIN->nb_dd_dofs()  ... nombre de dof dans un sous domaine
822   My_SUB_DOMAIN->give_dd_dof(i)->dof->rank() ... le rank du dof dans le sous domaine
823   My_SUB_DOMAIN->give_dd_dof(i)->shared  ... le nombre de sous domaines se
    partageant le Dof
824   My_SUB_DOMAIN->give_dd_dof(i)->global_rank  ... le rank du dof dans la structure
    globale
825   */
826
827      Timer_counter.stop_time("REDUCED_STATE::Initialize");
828   }
829   void REDUCED_STATE::polynomial_shape() // purely local
830   {
831      Timer_counter.start_time("REDUCED_STATE::Polynomial_shape");
832      if (Nb_Shape == 0)  {
833           VECTOR R(A_buff.n);
834           R = 1.;
835           subspace_expansion(R, 0.);
836      }
837      Timer_counter.stop_time("REDUCED_STATE::Polynomial_shape");
838   }
839   void REDUCED_STATE::init_problem() // purely local
840   {
841      Inc_last_selection = 0;
842      Inc_last_update_cov = 0;
843   }
844   void REDUCED_STATE::init_increment() // purely local
845   {
846      Timer_counter.start_time("REDUCED_STATE::Init_increment");
847      if (its_ROM->Nb_increments > 1) {
848         for (int i=0;i<Nb_Shape;i++) {
849            a_buff(i,its_ROM->Nb_increments-1) = a_buff(i,its_ROM->Nb_increments-2);
850            a_incr[i] = 0.;
```

```
850         }
851       }
852       else  {
853         for (int i=0;i<Nb_Shape;i++) {
854           a_buff(i,its_ROM->Nb_increments-1) = 0.;
855           a_incr[i] = 0.;
856         }
857       }
858       Timer_counter.stop_time("REDUCED_STATE::Init_increment");
859     }

      void REDUCED_STATE::init_cov_hist() // purely local
860   {
861     Timer_counter.start_time("REDUCED_STATE::Init_cov_hist");
862     if (Nb_Shape > 0) {
863       MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
864
865       cov *= its_ROM->gamma_history;
866     }
867     Timer_counter.stop_time("REDUCED_STATE::Init_cov_hist");
868   }

      void REDUCED_STATE::update_cov_hist() // purely local
869   {
870     Timer_counter.start_time("REDUCED_STATE::Update_cov_hist");
871     int  Nb_increments = its_ROM->Nb_increments;
872     if ((its_ROM->true_for_adaptive_reduction) && (Nb_increments >
      Inc_last_update_cov))  {

873       MATRIX a_dof(Nb_Shape,Nb_increments-
      Inc_last_update_cov,a_buff,0,Inc_last_update_cov);
874       MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
875
876       if (memory_factor < 0.9999) cov *= memory_factor; // for EDF
877       cov += a_dof*transpose(a_dof);
878
879       Inc_last_update_cov = Nb_increments;

880     }
881   /*
882     else if ((its_ROM->true_for_adaptive_reduction) && (Nb_increments ==
      Inc_last_update_cov)) {

883       MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
884
885       if (its_ROM->convergence_acceleration) {
886         cov *= its_ROM->gamma_history; // for EDF
887         for (int i=0;i<Nb_Shape;i++) {
888           for (int j=0;i<Nb_Shape;i++) {
889             cov(i,j) += a_incr[i]*a_incr[j]; // for EDF
890           }
891         }
892       }

893     }
894   */
895     Timer_counter.stop_time("REDUCED_STATE::Update_cov_hist");
896   }
```

```
899
      void REDUCED_STATE::attenuate_cov_hist() // purely local
900   {
901      Timer_counter.start_time("REDUCED_STATE::Attenuate_cov_hist");
902      if (its_ROM->true_for_adaptive_reduction) {
903       update_cov_hist();
904       MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
905       cov *= memory_factor;
906     }
907     Timer_counter.stop_time("REDUCED_STATE::Attenuate_cov_hist");
908   }
909
      LIST<int> REDUCED_STATE::sort(LIST<double>& beta) // purely local
910   {
911      Timer_counter.start_time("REDUCED_STATE::Sort");
912      LIST<int> Lindices(beta.size());
913      double beta_max;
914      int i_max;
915      int j_max;
916
917      for (int i=0; i<Lindices.size(); i++) Lindices[i] = i;
918
919      for (int k=0;k<Lindices.size()-1;k++) {
920         i_max = k;
921         beta_max = fabs(beta[Lindices[i_max]]);
922         for (int i=k+1;i<Lindices.size();i++) {
923            if (fabs(beta[Lindices[i]]) > beta_max) {
924               beta_max = fabs(beta[Lindices[i]]);
925               i_max = i;
926            }
927         }
928         j_max = Lindices[k];
929         Lindices[k] = Lindices[i_max];
930         Lindices[i_max] = j_max;
931      }
932      Timer_counter.stop_time("REDUCED_STATE::Sort");
933      return Lindices;
934   }
935
      void REDUCED_STATE::power_method(DMATRIX &Lambda, MATRIX &Phi) // purely local
936   {
937      Timer_counter.start_time("REDUCED_STATE::Power_method");
938
939     MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
940     double epsilon = 0.1*epsilon_pod;
941
942     if ( norm(cov-transpose(cov)) > 1.e-4*cov(0,0) )
943       ERROR("Error in REDUCED_STATE::power_method, the matrix must be symetric");
944     Phi.resize(cov.n,cov.n);
945     Lambda.resize(cov.n);
946
947     MATRIX Test(cov.n,cov.n);
948     MATRIX Y(cov.n,cov.n);
949     MATRIX R(cov.n,cov.n);
950     double norme_test;
951     LIST<double> beta(cov.n);
952     LIST<int> Renum;
953     MATRIX proj(cov.n,cov.n);
```

```
954     SMATRIX Phi_Phi;
955     double beta_max;
956
957     Phi = 0.;
958     for (int i=0; i<cov.n; i++) Phi(i,i) = 1.;
959     int max_its = cov.n * 10;
960     for (int its=0; its<max_its; its++) {
961       Test = Phi;
962
963   // orthogonalization
      //    for (int j=0;j<cov.n;j++) Test(j,0) = Phi(j,0); /// utilité ???? on a déja
      dit dans la ligne d'avant la meme chose !!!
964       for (int i=0;i<cov.n-1;i++){
965          MATRIX Test_orth(Phi.n,i+1,Test,0,0);
966          MATRIX Phi_col(Phi.n,1,Phi,0,i+1);
967
968          proj = Test_orth * transpose(Test_orth) ;
969          MATRIX newTest = Phi_col - proj* Phi_col;
970          norme_test = norm(newTest);
971          if (norme_test > 1.e-10) newTest /= norme_test;
972          else {
973            for (int j=0;j<cov.n;j++) newTest(j,0) = RANDOM_LAW::get_random_double();
974            newTest = newTest - proj* newTest;
975            norme_test = norm(newTest);
976            newTest /= norme_test;
977          }
978          for (int j=0;j<cov.n;j++) Test(j,i+1) = newTest(j,0);
979
980       }
981        Y = cov*Test;
982        beta_max = 0.;
983        for (int i=0;i<cov.n;i++) {
984          MATRIX Col(cov.n,1,Y,0,i);
985          beta[i] = norm(Col);
986          if (beta[i] > beta_max) beta_max = beta[i];
987        }
988        for (int i=0;i<cov.n;i++) {
989  //        if (beta[i] < epsilon*beta_max) {
990          if (beta[i] < 0.01*epsilon*beta_max) {
991            for (int j=0;j<cov.n;j++) Y(j,i) = Test(j,i);
992          }
993          else {
994            for (int j=0;j<cov.n;j++) Y(j,i) /= beta[i];
995          }
996        }
997        R = Y - Test;
998        Renum = sort(beta);
999        for (int i=0;i<cov.n;i++) {
1000         for (int j=0;j<cov.n;j++) Phi(j,i) = Y(j,Renum[i]);
1001       }
          if (norm(R) < epsilon * sqrt(double(cov.n)) )  break;
```

```
1002      }
1003
1004      for (int j=0;j<cov.n;j++) {
1005        Lambda(j,j) = beta[Renum[j]];
1006      }
1007
1008      Timer_counter.stop_time("REDUCED_STATE::Power_method");
1009    }

      void REDUCED_STATE::set_state(VECTOR& Y, double coef) // purely local
1010  {
1011     Timer_counter.start_time("REDUCED_STATE::Set_state");
1012     int  Nb_increments = its_ROM->Nb_increments;
1013
1014     if (Nb_increments > 1) {
1015     for (int i=0;i<A_buff.n;i++) {
1016        Y[i] = 0;
1017        for (int j=0;j<Nb_Shape;j++) {
1018          Y[i] += A_buff(i,j)*(coef*a_buff(j,Nb_increments-1)+(1.-coef)*a_buff
      (j,Nb_increments-2));
1019        }
1020      }
1021      }
1022      else  {
1023        for (int i=0;i<A_buff.n;i++) {
1024           Y[i] = 0;
1025           for (int j=0;j<Nb_Shape;j++) {
1026             Y[i] += A_buff(i,j)*coef*a_buff(j,Nb_increments-1);
1027           }
1028         }
1029      }
1030      Timer_counter.stop_time("REDUCED_STATE::Set_state");
1031    }

      void REDUCED_STATE::set_shape_fct(VECTOR& Y, int ishape) // purely local
1032  {
1033    Y.resize(A_buff.n);
1034    if (ishape<Nb_Shape)  {
1035      for (int i=0;i<A_buff.n;i++) {
1036         Y[i] = A_buff(i,ishape);
1037        }
1038      }
1039      else  {
1040        Y = 0.;
1041      }
1042    }

      void REDUCED_STATE::set_shape_dof_tot(MESH* mesh, int ishape) // ambigue !!!!
      almost local
1043  {
1044     if (ishape<Nb_Shape)  {
1045       for (int i=0;i<mesh->nb_dof();i++) {
1046           mesh->get_dof(i)->d_dof_tot = A_buff(i,ishape);
1047       }
1048     }
1049     else
1050       for (int i=0;i<mesh->nb_dof();i++)  mesh->get_dof(i)->d_dof_tot = 0.;
1051  }
1052
1053
1054
```

```
1055
      void REDUCED_STATE::set_dof_increment(MESH* mesh) // ambigue !!!! almost local
1056  {
1057     DOF* dof;
1058     for (int i=0;i<mesh->nb_dof();i++) {
1059        dof=mesh->get_dof(i);
1060        if  (dof->if_fixed == FALSE){
1061          dof->incr_dof = 0.;
1062          for (int j=0;j<Nb_Shape;j++) {
1063            dof->incr_dof += A_buff(i,j)*a_incr[j];
1064          }
1065        }
1066     }
1067
1068  }
      void REDUCED_STATE::set_dof_tot(MESH* mesh) // ambigue !!!! almost local
1069  {
1070     DOF* dof;
1071     for (int i=0;i<mesh->nb_dof();i++) {
1072        dof=mesh->get_dof(i);
1073        if  (dof->if_fixed == FALSE){
1074          dof->d_dof_tot = 0.;
1075          for (int j=0;j<Nb_Shape;j++) {
1076             dof->d_dof_tot += A_buff(i,j)*a_buff(j,its_ROM->Nb_increments-1);
1077          }
1078        }
1079     }
1080
1081  }
      void REDUCED_STATE::ortho_shape_fct() // parallele ready ( jamais utilsée dans le
      code - repond pas au grep ??? )
1082  {
1083     Timer_counter.start_time("REDUCED_STATE::Ortho_shape_fct");
1084     int  Nb_increments = its_ROM->Nb_increments;
1085     MATRIX A_ortho(A_buff.n,A_buff.m);
1086     int ishape;
1087     int jshape;
1088     int kshape;
1089     int i;
1090     double proj_contrib;
1091     VECTOR proj(1);
1092     VECTOR GLC_temp_g_sum(1);
1093     VECTOR A_ortho_col(A_ortho.n);
1094     double normAortho2;
1095
1096
1097     Out<<"ROM shape function orthogonalization"<<endl;
1098     Gene_last_orthogonalization = Basis_generation;

1099     kshape = 0;
1100     for (ishape=0;ishape<Nb_Shape;ishape++)  {
1101        for (i=0;i<A_ortho.n;i++) A_ortho(i,kshape) = A_buff(i,ishape);

1102        for (jshape=0;jshape<ishape;jshape++)  {
1103           proj = 0.;
           for (i=0;i<A_ortho.n;i++) {
```

```
1104                proj_contrib = A_ortho(i,kshape) * A_buff(i,jshape);
1105                if (its_ROM->reference_problem->Is_parallel_computation &&
      is_dof_basis && is_global_basis)  proj_contrib /= Shared[i];
1106                proj[0] += proj_contrib;
1107             }
1108             if (its_ROM->reference_problem->Is_parallel_computation &&
1109   is_global_basis)  glc.g_sum(proj);

                 for (i=0;i<A_ortho.n;i++) {
1110                A_ortho(i,kshape) -= proj[0] * A_buff(i,jshape);
1111                A_ortho_col[i] = A_ortho(i,kshape);
1112             }
1113          }

             if(its_ROM->reference_problem->Is_parallel_computation && is_global_basis) {
1115             if (is_dof_basis)   {
1116                for (int i=0; i<A_ortho_col.size(); i++) A_ortho_col[i] = A_ortho_col
      [i]/ Shared[i] ;
1117             }
1118             GLC_temp_g_sum[0] = norm(A_ortho_col);
1119             GLC_temp_g_sum[0]  *= GLC_temp_g_sum[0];
1120             glc.g_sum(GLC_temp_g_sum);
1121             normAortho2 = GLC_temp_g_sum[0];
1122          } else {
1123                normAortho2 = norm(A_ortho_col);
1124                normAortho2 *= normAortho2;
1125          }


1127          if (normAortho2 > 1.e-10) {
1128             normAortho2 = sqrt(normAortho2);
1129             for (i=0;i<A_ortho.n;i++) A_ortho(i,kshape) /= normAortho2;
1130             kshape +=1;
1131          }
1132       }


1134       MATRIX V(kshape,Nb_Shape);
1135       double V_contrib;
1136       for (ishape=0;ishape<kshape;ishape++) {
1137          for (jshape=0;jshape<Nb_Shape;jshape++) {
1138             V(ishape,jshape) = 0.;
1139             for (i=0;i<A_ortho.n;i++) {
1140                V_contrib = A_ortho(i,ishape) * A_buff(i,jshape);
1141                if (its_ROM->reference_problem->Is_parallel_computation &&
      is_dof_basis && is_global_basis)  V_contrib /= Shared[i];
1142                V(ishape,jshape) += V_contrib;
1143             }
1144          }
1145       }
1146       if (its_ROM->reference_problem->Is_parallel_computation && GG_tot.size()>0 &&
1147   is_global_basis)  glc.g_sum(V);


1148       MATRIX a(Nb_Shape,Nb_increments,a_buff,0,0);
1149       a = V*a;

          MATRIX Cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
```

```
1151      MATRIX Cov2(kshape,kshape);
1152      Cov2 = V * ( Cov * transpose(V) );
1153      Nb_Shape = kshape;
1154      MATRIX Cov3(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
1155      Cov3 = Cov2;
1156
1157      MATRIX A(A_buff.n,Nb_Shape,A_buff,0,0);
1158      MATRIX A2(A_buff.n,Nb_Shape,A_ortho,0,0);
1159      A = A2;
1160
1161      Timer_counter.stop_time("REDUCED_STATE::Ortho_shape_fct");
1162  }
1163
1164  void REDUCED_STATE::init_shape(int k)
1165  {
1166    for (int i=0;i<A_buff.n;i++) A_buff(i,k) = 0.;
1167  }
1168
1169  void REDUCED_STATE::give_global_A()
1170  {
1171     Zfstream file;
1172     STRING A_file;
1173
1174  //        MATRIX A_buff; la matrice a assembler
1175
1176     if (its_ROM->reference_problem->Is_parallel_computation && is_global_basis){
1177        MATRIX Global_A(My_SUB_DOMAIN->global_dof_nb,Nb_Shape);
1178        Global_A = 0;
1179
1180        if (is_dof_basis) {
1181
1182        for (int i=0;i<A_buff.n;++i)
1183          Out<<" local : "<<My_SUB_DOMAIN->give_dd_dof(i)->dof->rank()<< " ,
1184  global : "<<My_SUB_DOMAIN->give_dd_dof(i)->global_rank<<endl;
1185
1186           for (int jshape = 0; jshape<Nb_Shape; jshape++) {
1187               for (int i=0;i<A_buff.n;++i)
1188                    Global_A(My_SUB_DOMAIN->give_dd_dof(i)->global_rank,jshape)=
1189  A_buff(i,jshape)/Shared[i];
1190           }
1191           glc.g_sum(Global_A);
1192
1193
1194           A_file = "matrice_A"+itoa(The_mp_client.my_node);
1195  /*
1196           file.open(A_file(),ios::out);
1197           file<<Global_A.n<<"\t"<<Nb_Shape<<endl;
1198           for (int i=0;i<Global_A.n;i++) {
1199              for (int j=0;j<Nb_Shape;j++)  {
1200                  file<<Global_A(i,j)<<"\t";
1201              }
1202              file<<endl;
1203           }
1204           file.close();
1205  */
1206           file.open(A_file(),ios::out);
1207           file<<A_buff.n<<"\t"<<Nb_Shape<<endl;
1208           for (int i=0;i<A_buff.n;i++) {
```

```
1200                for (int j=0;j<Nb_Shape;j++)  {
1201                    file<<A_buff(i,j)<<"\t";
1202                }
1203                file<<endl;
1204             }
1205             file.close();
1206
1207          } else {
1208
1209
1210          }
1211       }else{
1212          if (is_dof_basis) {
1213
1214             A_file = "matrice_A";
1215
1216             file.open(A_file(),ios::out);
1217             file<<A_buff.n<<"\t"<<Nb_Shape<<endl;
1218             for (int i=0;i<A_buff.n;i++) {
1219                for (int j=0;j<Nb_Shape;j++)  {
1220                    file<<A_buff(i,j)<<"\t";
1221                }
1222                file<<endl;
1223             }
1224             file.close();
1225
1226          } else {
1227
1228
1229
1230          }
1231       }
1232
1233    }
1234
1235    void REDUCED_STATE::Eigenproblem(DMATRIX &Lambda, MATRIX &Phi)
1236    {
1237
1238     /*
1239       This is derived from the Algo procedures, by Bowdler, Martin, Reinsch,
1240       and Wilkinson, Handbook for Auto. Comp., Vol.ii-Linear Algebra,
1241       and the corresponding subroutine in EISPACK.
1242
1243                Computes all eigenvalues and eigenvectors of a real symmetric matrix
1244    by
1245                reduction to tridiagonal form followed by QL iteration.
1246                Lambda = the diagonal eigenvalues matrix
1247                Phi = the eigenvectors matrix
1248     */
1249
1250       Timer_counter.start_time("REDUCED_STATE::Eigenproblem");
1251
1252       MATRIX cov(Nb_Shape,Nb_Shape,Covariance_history_buff,0,0);
1253
1254       if ( norm(cov-transpose(cov)) > 1.e-4*cov(0,0) )
```

```
1242          ERROR("Error in REDUCED_STATE::Eigenproblem, the matrix must be symetric");
1243      Phi.resize(cov.n,cov.n);
1244      Lambda.resize(cov.n);

          VECTOR d;
1246      d.resize(cov.n);
1247      VECTOR e;
1248      e.resize(cov.n);

          // Store matrix cov to Phi
1250      for (int i = 0; i < cov.n; i++) {
1251         for (int j = 0; j < cov.n; j++) {
1252               Phi(i,j) = cov(i,j);
1253         }
1254      }

          // Tridiagonalize.
1256  /*
1257               Householder reduction of a real symmetric matrix Phi[0..n-1][0..n-1].
      (The input matrix cov
1258               is stored in Phi.) On output, Phi is replaced by the orthogonal matrix
      Q effecting
1259               the transformation. d[0..n-1] contains the diagonal elements of the
      tridiagonal matrix and
1260               e[0..n-1] the off-diagonal elements, with e[0]=0.
1261  */
1262      tred2(cov.n, d, e, Phi);

          // Diagonalize.
1264  /*
1265               QL algorithm, to determine the eigenvalues and eigenvectors of a real
      symmetric matrix previously
1266               reduced by tred2. On input, d[1..n] contains the diagonal elements of
      the tridiagonal matrix.
1267               On output, it returns the eigenvalues. The vector e[1..n] inputs the
      subdiagonal elements of the
1268               tridiagonal matrix, with e[1] arbitrary. On output e is destroyed. If
      the eigenvectors of a matrix
1269               that has been reduced by tred2 are required, then Phi is input as the
      matrix output by tred2.
1270               The k_th column of Phi returns the normalized eigenvector
      corresponding to d[k]
1271  */
1272      tql2(cov.n, d, e, Phi);

       // updating Lambda
1274      for (int j=0;j<cov.n;j++) {
1275            Lambda(j,j) = d[j];
1276      }
1277      Timer_counter.stop_time("REDUCED_STATE::Eigenproblem");

1279  }

      //
      ****************************************************************************//
1280  void REDUCED_STATE::tred2(int n, VECTOR &d, VECTOR &e, MATRIX &V )
1281  {
```

```
         for (int j = 0; j < n; j++) {
1283        d[j] = V(n-1,j);
1284     }
1285
         // Householder reduction to tridiagonal form.
1286
1287     for (int i = n-1; i > 0; i--) {
1288
1289        // Scale to avoid under/overflow.
1290
1291        double scale = 0.0;
1292        double h = 0.0;
1293        for (int k = 0; k < i; k++) {
1294           scale += fabs(d[k]);
1295        }
1296        if (scale == 0.0) {
1297           e[i] = d[i-1];
1298           for (int j = 0; j < i; j++) {
1299              d[j] = V(i-1,j);
1300              V(i,j) = 0.0;
1301              V(j,i) = 0.0;
1302           }
1303        } else {
1304
1305           // Generate Householder vector.
1306
1307           for (int k = 0; k < i; k++) {
1308              d[k] /= scale;
1309              h += d[k] * d[k];
1310           }
1311           double f = d[i-1];
1312           double g = sqrt(h);
1313           if (f > 0) {
1314              g = -g;
1315           }
1316           e[i] = scale * g;
1317           h = h - f * g;
1318           d[i-1] = f - g;
1319           for (int j = 0; j < i; j++) {
1320              e[j] = 0.0;
1321           }
1322
1323           // Apply similarity transformation to remaining columns.
1324
1325           for (int j = 0; j < i; j++) {
1326              f = d[j];
1327              V(j,i) = f;
1328              g = e[j] + V(j,j) * f;
1329              for (int k = j+1; k <= i-1; k++) {
1330                 g += V(k,j) * d[k];
1331                 e[k] += V(k,j) * f;
1332              }
1333              e[j] = g;
1334           }
1335           f = 0.0;
1336           for (int j = 0; j < i; j++) {
1337              e[j] /= h;
1338              f += e[j] * d[j];
```

```
            }
            double hh = f / (h + h);
            for (int j = 0; j < i; j++) {
               e[j] -= hh * d[j];
            }
            for (int j = 0; j < i; j++) {
               f = d[j];
               g = e[j];
               for (int k = j; k <= i-1; k++) {
                  V(k,j) -= (f * e[k] + g * d[k]);
               }
               d[j] = V(i-1,j);
               V(i,j) = 0.0;
            }
         }
         d[i] = h;
      }

      // Accumulate transformations.

      for (int i = 0; i < n-1; i++) {
         V(n-1,i) = V(i,i);
         V(i,i) = 1.0;
         double h = d[i+1];
         if (h != 0.0) {
            for (int k = 0; k <= i; k++) {
               d[k] = V(k,i+1) / h;
            }
            for (int j = 0; j <= i; j++) {
               double g = 0.0;
               for (int k = 0; k <= i; k++) {
                  g += V(k,i+1) * V(k,j);
               }
               for (int k = 0; k <= i; k++) {
                  V(k,j) -= g * d[k];
               }
            }
         }
         for (int k = 0; k <= i; k++) {
            V(k,i+1) = 0.0;
         }
      }
      for (int j = 0; j < n; j++) {
         d[j] = V(n-1,j);
         V(n-1,j) = 0.0;
      }
      V(n-1,n-1) = 1.0;
      e[0] = 0.0;
   }

//
*******************************************************************************//
void REDUCED_STATE::tql2 (int n, VECTOR &d, VECTOR &e, MATRIX &V )
{


      for (int i = 1; i < n; i++) {
         e[i-1] = e[i];
```

```
1394         }
1395         e[n-1] = 0.0;
1396
1397         double f = 0.0;
1398         double tst1 = 0.0;
1399         double eps = pow(2.0,-52.0);
1400         for (int l = 0; l < n; l++) {
1401
1402            // Find small subdiagonal element
1403
1404            tst1 = Zmax(tst1,fabs(d[l]) + fabs(e[l]));
1405            int m = l;
1406
1407           // Original while-loop from Java code
1408            while (m < n) {
1409               if (fabs(e[m]) <= eps*tst1) {
1410                  break;
1411               }
1412               m++;
1413            }
1414
1415
1416            // If m == l, d[l] is an eigenvalue,
1417            // otherwise, iterate.
1418
1419            if (m > l) {
1420               int iter = 0;
1421               do {
1422                  iter = iter + 1;  // (Could check iteration count here.)
1423
1424                  // Compute implicit shift
1425
1426                  double g = d[l];
1427                  double p = (d[l+1] - g) / (2.0 * e[l]);
1428                  double r = hypot(p,1.0);
1429                  if (p < 0) {
1430                     r = -r;
1431                  }
1432                  d[l] = e[l] / (p + r);
1433                  d[l+1] = e[l] * (p + r);
1434                  double dl1 = d[l+1];
1435                  double h = g - d[l];
1436                  for (int i = l+2; i < n; i++) {
1437                     d[i] -= h;
1438                  }
1439                  f = f + h;
1440
1441                  // Implicit QL transformation.
1442
1443                  p = d[m];
1444                  double c = 1.0;
1445                  double c2 = c;
1446                  double c3 = c;
1447                  double el1 = e[l+1];
1448                  double s = 0.0;
1449                  double s2 = 0.0;
1450                  for (int i = m-1; i >= l; i--) {
1451                     c3 = c2;
```

```
                    c2 = c;
                    s2 = s;
                    g = c * e[i];
                    h = c * p;
                    r = hypot(p,e[i]);
                    e[i+1] = s * r;
                    s = e[i] / r;
                    c = p / r;
                    p = c * d[i] - s * g;
                    d[i+1] = h + s * (c * g + s * d[i]);

                    // Accumulate transformation.

                    for (int k = 0; k < n; k++) {
                       h = V(k,i+1);
                       V(k,i+1) = s * V(k,i) + c * h;
                       V(k,i) = c * V(k,i) - s * h;
                    }
                 }
                 p = -s * s2 * c3 * el1 * e[l] / dl1;
                 e[l] = s * p;
                 d[l] = c * p;

                 // Check for convergence.

              } while (fabs(e[l]) > eps*tst1);
           }
           d[l] = d[l] + f;
           e[l] = 0.0;
        }

        // Sort eigenvalues and corresponding vectors.

        for (int i = 0; i < n-1; i++) {
           int k = i;
           double p = d[i];
           for (int j = i+1; j < n; j++) {
              if (d[j] > p) {
                     // modified sort
                 k = j;
                 p = d[j];
              }
           }
           if (k != i) {
              d[k] = d[i];
              d[i] = p;
              for (int j = 0; j < n; j++) {
                 p = V(j,i);
                 V(j,i) = V(j,k);
                 V(j,k) = p;
              }
           }
        }
     }
```

Annexe 3

Computational Materials Science 50 (2011) 1597–1605

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

A robust adaptive model reduction method for damage simulations

D. Ryckelynck *, D. Missoum Benziane, S. Cartel, J. Besson

MINES ParisTech, Centre des matériaux, CNRS UMR 7633, BP 87 91003 Evry Cedex, France

ARTICLE INFO

Article history:
Received 18 October 2010
Accepted 29 November 2010
Available online 12 January 2011

Keywords:
APHR method
POD
Separated representation
Continuum damage
Bifurcation

ABSTRACT

In our opinion, many of complex numerical models in materials science can be reduced without losing their physical sense. Due to solution bifurcation and strain localization of continuum damage problems, damage predictions are very sensitive to any model modification. Most of the robust numerical algorithms intend to forecast one approximate solution of the continuous model despite there are multiple solutions. Some model perturbations can possibly be added to the finite element model to guide the simulation toward one of the solutions. Doing a model reduction of a finite element damage model is a kind of model perturbation. If no quality control is performed the prediction of the reduced-order model (ROM) can really differ from the prediction of the full finite element model. This can happen using the snapshot Proper Orthogonal Decomposition (POD) model reduction method. Therefore, if the expected purpose of the reduced approximation is to estimate the solution that the finite element simulation should give, an adaptive reduced-order modeling is required when reducing finite element damage models.

We propose an adaptive reduced-order modeling method that enables to estimate the effect of loading modifications. The Rousselier continuum damage model is considered. The differences between the finite element prediction and the one provided by the adapted reduced-order model (ROM) remain stable although various loading perturbations are introduced. The adaptive algorithm is based on the APHR (A Priori Hyper Reduction) method. This is an incremental scheme using a ROM to forecast an initial guess solution to the finite element equations. If, at the end of a time increment, this initial prediction is not accurate enough, a finite element correction is added to the ROM prediction. The proposed algorithm can be viewed as a two step Newton–Raphson algorithm. During the first step the prediction belongs to the functional space related to the ROM and during the second step the correction belongs to the classical FE functional space. Moreover the corrections of the ROM predictions enable to expand the basis related to the ROM. Therefore the ROM basis can be improved at each increment of the simulation. The efficiency of the adaptive algorithm is checked comparing the amount of global linear solutions involved in the proposed scheme versus the amount of global linear solutions involved in the classical incremental Newton–Raphson scheme. The quality of the proposed approximation is compared to the one provided by the classical snapshot Proper Orthogonal Decomposition (POD) method.

1. Introduction

Continuum damage elasto-plastic models are widely used for fatigue life prediction or crack growth on metallic components [1–3]. Due to the bifurcation of the solutions, the damage models are very sensitive to any model modification. Nonlocal models have been proposed to avoid the mesh dependency of the solution assuming the element size is small enough. For more details about mesh nonlocal models, we refer the reader to [4]. Various approaches are available to study the bifurcation modes related to the continuum damage models [5,4,6–8]. In this paper we assume that a numerical simulation aims to estimate one of the admissible solutions of the nonlinear continuous model. Only one prediction must be forecast per simulation. Model perturbations are introduced to obtain various predictions of the mechanical state. The reduced basis approximations introduce errors that can be amplified during the bifurcation of the solution. This paper aims to show that the APHR (A Priori Hyper Reduction) method proposed in [9] enables to build robust reduced approximations for continuum damage models based on Rousselier constitutive law [10,11]. This reduced approximation is a time/space separated representation of the displacement, which is defined over the whole time interval of the simulation and the entire domain.

The development of large FE models increases the need of low order models created by model reduction methods. The availability of reduced-order models (ROMs) can greatly facilitate the solution of series of mechanical problems appearing in optimization problems for instance. The formulation of the reduced-equations differs from the formulation of the detailed equations in the choice of the

* Corresponding author.
E-mail address: david.ryckelynck@mines-paristech.fr (D. Ryckelynck).

0927-0256/$ - see front matter © 2011 Published by Elsevier B.V.
doi:10.1016/j.commatsci.2010.11.034

functional space related to the primal state variables. Most of the time, the state variables are considered as a linear combination of known fields. In case of a posteriori approach, these fields are obtained by solving detailed preliminary problems. In the framework of modal superposition method, these fields are the eigenvectors related to the free-vibration problem. If the preliminary problems provide linearly independent fields, then the coefficients of the linear combination are the reduced-state variables of the ROM. On the contrary, one has to build a reduced-basis of the subspace spanned by the known fields. The Proper Orthogonal Decomposition (POD) aims to create such a basis from a set of time dependent fields. The POD model reduction method has been used in a wide range of nonlinear incremental simulations and optimization problems in fluid mechanics [12], in materials science [13,14], in thermal science [15] and real-time surgery simulation [16]. This method originates from the Karhunen–Loève expansion [17,18] developed for statistical analyses. A POD basis is an optimal basis of the state subspace spanned by forecast states possibly related to various simulations of the response of the detailed model. The optimality of the POD has been proved in [19]. The snapshot POD proposed by Sirovich [20] has had an important contribution to the development of POD basis for the reduction of nonlinear time-dependent models. This method reduces the size of the eigenproblem involved in the construction of the POD vectors. Some details about this method are given in Section 4. The main drawback of the classical POD and of the snapshot POD methods is the lack of adaptive procedure in order to modify the ROM basis according to an error indicator. During the same period, Ladevèze proposed a competitive decomposition method dedicated to elasto-plastic models named the radial loading decomposition (RLD) [21]. This decomposition is performed by the LATIN method [21] without solving the full incremental detailed model. The LATIN method aims to create iteratively a linear problem to forecast the response of the nonlinear detailed model. This linear problem is defined over the entire time interval. Therefore, a time and space separated representation enables to define the decomposition of the forecast state evolution. Various LATIN algorithms were proposed to solve optimization problems [22,23] and multiscale problems [24]. The similarities between the RLD and the POD have been stated in [25]. LATIN algorithms can control the quality of the separated representation using a non incremental solution scheme. But the extension of such non incremental scheme to any constitutive equations is not straightforward. For this reason, the APHR method has been proposed in [9]. This is an adaptive model reduction method using a classical incremental scheme. The extension of the reduced basis can be performed using a Krylov subspace as proposed in [9], or using a known state evolution [26], or using nonlinear global corrections as proposed in this paper. The nonlinear corrections are introduced using a multi-level approximation. The first level is the reduced approximation having global reduced-state variables. The second level is the finite element correction having nodal variables. The proposed adaptive method is coined the multi-level APHR method. Despite the solution of damage problem is not unique in case of bifurcation, robust integration schemes enable to forecast one mechanical state per finite element simulation. For details on robust integration of finite element damage problems the reader can refer to [27]. The purpose of a reduced approximation is to forecast the finite element solution using less variables and less equations. But the reduced approximation being a modification of the finite element model, the approximation errors can be amplified during bifurcation. This paper aims to show that reduced approximation being adapted, the distance between the FE prediction and the APHR prediction is almost constant during the bifurcation of the solution.

The paper is organized in the following manner. Section 2 introduces the formulation of the mechanical problem. The functional spaces related to the finite element problem and to the POD approximation are described in this section. The formulation of the multi-level approximation and the adaptive algorithm related to the APHR method are detailed in Section 3. Section 4 reports the numerical results elucidating the robustness of the proposed method. The amount of linear solutions required during the APHR simulation is compared to the amount of linear solutions required to forecast the finite element solution. Section 5 is the conclusion of this paper.

2. Formulation of the continuous model

The continuous model of concern is a parametrized mechanical model. We denote $\{\mathbf{p}\}$ the column of the model parameters. These parameters can be related to design choices, material coefficients or loading conditions. We consider a series of mechanical problems related to a series of parameter values $(\{\mathbf{p}\}_\alpha)_{\alpha=1\ldots N}$. In this paper, we consider parameters related to loading perturbations.

The detailed model is described using the finite strain formalism. At time instant t, the continuous medium is occupying a domain Ω. The nonlinear system is analyzed over a time interval $]0,T]$. The reference configuration is denoted Ω^0. It can be the domain either at time instant $t=0$ (total Lagrangian formulation) or at time instant t (update Lagrangian formulation). The displacement field at time t is defined on Ω^0 and it is denoted by $\mathbf{u}(\mathbf{X}, t, \{\mathbf{p}\}_\alpha)$. $\mathbf{X}$ is denoting the initial position of a material point in Ω^0. The second Piola–Kirchhoff stress tensor $\mathbf{S}$ is a nonlinear function of the deformation gradient history:

$$\mathbf{S} = \boldsymbol{\Sigma}(\mathbf{F}_\tau, \tau \leqslant t) \tag{1}$$

where $\boldsymbol{\Sigma}$ is a formal operator that must be defined by constitutive equations and $\mathbf{F}_\tau$ is the deformation gradient at time instant τ. We observe that the following general relations hold:

$$F_{ij} = \delta_{ij} + \frac{\partial u_i}{\partial X_j} \tag{2}$$

$$\boldsymbol{\sigma} = \frac{1}{det(\mathbf{F})}\mathbf{F}\cdot\mathbf{S}\cdot\mathbf{F}^T \tag{3}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor and δ_{ij} is the Kronecker delta. The Eulerian strain rate is :

$$\mathbf{D} = \mathrm{sym}(\dot{\mathbf{F}}\cdot\mathbf{F}^{-1}), \quad \text{with } \mathrm{sym}(\mathbf{C}) = \frac{1}{2}(\mathbf{C}+\mathbf{C}^T) \tag{4}$$

The boundary $\partial\Omega^0$ of Ω^0 is denoted by $\partial_U\Omega^0 \cup \partial_f\Omega^0$. On $\partial_U\Omega^0$, there is the Dirichlet condition $\mathbf{u}(\cdot, t, \{\mathbf{p}\}_\alpha) = \mathbf{u}_c(\cdot, t, \{\mathbf{p}\}_\alpha)$ for all t, where $\mathbf{u}_c$ is a given parametrized displacement field. On $\partial_f\Omega^0$, there is a given force field $\mathbf{f}(\cdot, t)$ depending on time t. The displacement field belongs to an affine function space $\mathcal{U}$ defined by:

$$\mathcal{U} = \left\{\mathbf{u}(\cdot, t, \{\mathbf{p}\}_\alpha) \in H^1(\Omega^0) | \mathbf{u}(\mathbf{X}, t, \{\mathbf{p}\}_\alpha) = \mathbf{u}_c(\mathbf{X}, t) \quad \forall \mathbf{X} \in \partial_U\Omega^0\right\} \tag{5}$$

The vector space associated to $\mathcal{U}$ is denoted $\mathcal{V}$. It does not depend on any parameter:

$$\mathcal{V} = \{\mathbf{u}(\cdot, t, \{\mathbf{p}\}_\alpha) \in H^1(\Omega^0) | \mathbf{u}(\mathbf{X}, t, \{\mathbf{p}\}_\alpha) = 0 \quad \forall \mathbf{X} \in \partial_U\Omega^0\} \tag{6}$$

The statement of the mechanical problem is the following. We want to find an estimation of the displacement field $\mathbf{u} \in \mathcal{U}$ defined by the constitutive equations and the principle of virtual work:

$$\int_{\Omega^0} \boldsymbol{\epsilon}(\mathbf{u}^*, \mathbf{u}) : \boldsymbol{\Sigma}(\mathbf{F}(\mathbf{u}), \tau \leqslant t)\, d\Omega^0 - \int_{\partial_f\Omega^0} \mathbf{u}^* \cdot \mathbf{f}(\mathbf{X}, t; \{\mathbf{p}\}_\alpha)\, d\Gamma^0 = 0 \quad \forall \mathbf{u}^* \in \mathcal{V} \tag{7}$$

where $\mathbf{u}^*$ is a test function and $\boldsymbol{\epsilon}$ is linear function of $\mathbf{u}^*$ such that :

$$\epsilon_{ij}(\mathbf{u}^*,\mathbf{u}) = \frac{1}{2}\left(\frac{\partial u_i^*}{\partial X_j} + \frac{\partial u_j^*}{\partial X_i} + \sum_{k=1}^{k=3}\left(\frac{\partial u_k^*}{\partial X_i}\frac{\partial u_k}{\partial X_j} + \frac{\partial u_k}{\partial X_i}\frac{\partial u_k^*}{\partial X_j}\right)\right) \tag{8}$$

According to the framework of the irreversible thermodynamic processes, a constitutive law can be defined by a choice of: internal variables, the Helmholtz free energy and a pseudo potential of dissipation [28,29] or complementary evolution equations. Examples of elasto-plastic or elastoviscoplastic constitutive models can be found in [30].

In this paper we consider the Rousselier model introduced in [10,11]. This is an elasto-plastic law with isotropic hardening to describe ductile fracture of metals. The damage evolution is related to the growth of cavities. As ductile rupture always involves large deformation levels, a finite strain model is required. The model used in this work is based on a the definition of corotational frame [31]. This relatively simple treatment of large deformation allows to keep a "small deformation" formalism for the constitutive equations. The velocity gradient $\mathbf{L} = \dot{\mathbf{F}}\cdot\mathbf{F}^{-1}$ is splitted into a symmetric part ($\mathbf{D}$) and a skew-symmetric part ($\mathbf{W}$). The rotation defining the corotational frame is defined by the following equation:

$$\dot{\mathbf{Q}}_c = \mathbf{W}\cdot\mathbf{Q}_c \quad \text{with} \quad \mathbf{Q}_c(t=0) = \mathbf{1} \tag{9}$$

Rotated stress ($\boldsymbol{\sigma}_c = \mathbf{Q}_c\cdot\boldsymbol{\sigma}\cdot\mathbf{Q}_c^T$) and deformation rate ($\dot{\boldsymbol{\varepsilon}}_c = \mathbf{Q}_c\cdot\mathbf{D}\cdot\mathbf{Q}_c^T$) tensors are used to express constitutive equations. The corresponding objective rate for the Cauchy stress tensor is the Jauman rate. An additive decomposition for $\dot{\boldsymbol{\varepsilon}}_c$ is used:

$$\dot{\boldsymbol{\varepsilon}}_c = \dot{\boldsymbol{\varepsilon}}_p + \dot{\boldsymbol{\varepsilon}}_e \tag{10}$$

where $\dot{\varepsilon}_p$ is the plastic strain rate tensor and $\dot{\boldsymbol{\varepsilon}}_e$ the elastic strain rate tensor given by $\dot{\boldsymbol{\varepsilon}}_e = \mathbf{E}:\dot{\boldsymbol{\sigma}}_c$ (hypoelasticity). $\mathbf{E}$ is Hooke's tensor. The internal variables are: $\boldsymbol{\varepsilon}_e$ and the cumulated plastic strain p.

The plastic yield surface is expressed as:

$$\Phi = \frac{\bar{\sigma}_c}{1-f} + \sigma_1 f D\exp\left(\frac{\mathrm{tr}(\boldsymbol{\sigma}_c)}{3(1-f)\sigma_1}\right) - \sigma_f(p) \tag{11}$$

where $\bar{\sigma}_c$ is the von Mises invariant of $\boldsymbol{\sigma}_c$, $\mathrm{tr}(\boldsymbol{\sigma}_c)$ is the trace of the rotated stress tensor and $\sigma_f(p)$ the flow stress of the undamaged material ; it depends on the isotropic hardening variable $p\cdot f$ is the damage variable corresponding to the void volume fraction (or porosity). σ_1 and D are model parameters. Assuming the normality rule, the plastic strain rate tensor is expressed as:

$$\dot{\boldsymbol{\varepsilon}}_p = (1-f)\dot{p}\frac{\partial\Phi}{\partial\boldsymbol{\sigma}_c} \tag{12}$$

The plastic multiplier $\dot{p}$ is expressed as $\dot{p} = \sqrt{\frac{2}{3}\dot{\boldsymbol{\varepsilon}}_p':\dot{\boldsymbol{\varepsilon}}_p'}$ where $\dot{\boldsymbol{\varepsilon}}_p'$ is the deviator of $\dot{\boldsymbol{\varepsilon}}_p$. Damage evolution is given using mass conservation so that:

$$\dot{f} = (1-f)\mathrm{tr}(\dot{\boldsymbol{\varepsilon}}_p) \tag{13}$$

Nucleation is simply modelled by a given initial porosity f_0. There is no continuous nucleation in this model. Coalescence of cavities is roughly modelled through strain localization. In spite of these assumptions, the Rousselier model is able to describe complex structural damage [8]. A nonlocal reformulation of the Rousselier model has been proposed in [32] to improve the robustness of the numerical integration and avoid mesh dependency caused by strain-softening.

2.1. The reference model

Using the finite element method [33], a backward Euler method and the Newton–Raphson algorithm, one can forecast different states of the system at different time instants. The purpose of is paper is not to explain how to get these states. Details on the finite element simulation of damage models can be found in [8] for instance. This section aims to recall some properties of the states forecast by a finite element model because it is the detailed model to reduce. Let us denote $\mathcal{U}_h$ and $\mathcal{V}_h$ the functional affine space and the vector space related to the finite element model. The finite element model involves n degrees of freedom $(q_j(t))_{j=1\ldots n}$ such that:

$$\mathcal{U}_h = \{\mathbf{u}\in\mathcal{V}|\exists\{\mathbf{q}\}\in\Re^n \tag{14}$$

$$\mathbf{u}(\mathbf{X},t,\{\mathbf{p}\}_\alpha) = \sum_{j=1}^{j=n}\mathbf{N}_j(\mathbf{X})\hat{q}_j(t,\{\mathbf{p}\}_\alpha) + \sum_{j=1}^{j=n_c}\widetilde{\mathbf{N}}_j(\mathbf{X})g_j(t) \tag{15}$$
$$\forall\mathbf{X}\in\Omega^0 \quad \forall t\in]0,T]\}$$

The shape functions $(\widetilde{\mathbf{N}}_j)_{j=1,\ldots n_c}$ are connected to the nodes belonging to $\partial_U\Omega^0$. The coefficient g_j are given displacement related to $\mathbf{u}_c$. Therefore, the given part of finite element displacement field is $\mathbf{u}_{ch}$ such that:

$$\mathbf{u}_{ch}(\mathbf{X},t) = \sum_{j=1}^{j=n_c}\widetilde{\mathbf{N}}_j(\mathbf{X})g_j(t) \tag{16}$$

The related vector space is such that:

$$\mathcal{V}_h = \mathrm{span}\{\mathbf{N}_1,\ldots,\mathbf{N}_n\} \tag{17}$$

Replacing $\mathcal{U}$ by $\mathcal{U}_h$ and $\mathcal{V}$ by $\mathcal{V}_h$ in the continuous formulation (7) we obtain the reference problem to solve: find $\mathbf{u}\in\mathcal{U}_h$ defined by the constitutive equations and the principle of virtual work:

$$\int_{\Omega^0}\boldsymbol{\epsilon}(\mathbf{u}^*,\mathbf{u}):\boldsymbol{\Sigma}(\mathbf{F}(\mathbf{u}),\tau\leqslant t)\,d\Omega^0 - \int_{\partial_f\Omega^0}\mathbf{u}^*\cdot\mathbf{f}(\mathbf{X},t;\{\mathbf{p}\}_\alpha)\,d\Gamma^0 = 0 \quad \forall\mathbf{u}^*\in\mathcal{V}_h \tag{18}$$

where $\boldsymbol{\Sigma}(\mathbf{F}(\mathbf{u}),\tau\leqslant t)$ is the second Piola–Kirchhoff stress tensor implicitely defined by the constitutive Eqs. (3), (4), (9)–(13).

Removing the imposed part $\mathbf{u}_{ch}$ of the forecast displacement fields, we are able to obtain a series of continuous displacements $\mathbf{U}_i\in\mathcal{V}_h$ for $i=1\ldots m$. The displacement fields $\mathbf{U}_i$ can be related to various parameter values $\{\mathbf{p}\}_\alpha$ and various time instants.

2.2. The POD reduced approximation

The main objective of the model reduction methods is to replace $\mathcal{U}_h$ and $\mathcal{V}_h$ by small dimensional functional subspaces in order to approximate $\mathbf{u}(\cdot,t,\{\mathbf{p}\}_\alpha)$ using previous results of simulations. The purpose of the snapshot POD is to use the list of the forecast displacement fields $(\mathbf{U}_i)_{i=1\ldots m}$ to define a reduced basis of the subspace span by these fields. Let's denote $(\boldsymbol{\psi}_k)_{k=1\ldots\hat{m}}$ the vectors of the reduced basis (in practice $\hat{m}<n$). Let us denote $\mathcal{V}_{ROM}$ the vector space related to the reduced-order model:

$$\mathcal{V}_{ROM} = \mathrm{span}\{\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_{\hat{m}}\} \tag{19}$$

Later, new state estimations related to modifications of the model can be computed using the following reduced approximation:

$$\mathbf{u}(\mathbf{X},t,\{\mathbf{p}\}_\alpha) = \sum_{k=1}^{k=\hat{m}}\boldsymbol{\psi}_k(\mathbf{X})a_k(t,\{\mathbf{p}\}_\alpha) + \mathbf{u}_{ch}(\mathbf{X},t), \quad \forall\mathbf{X}\in\Omega^0 \quad \forall t\in]0,T]. \tag{20}$$

The factors $(a_k)_{k=1\ldots\hat{m}}$ are the reduced-state variables of the ROM. These are global variables. The affine space related to the reduced basis approximation (20) is denoted $\mathcal{U}_{ROM}$:

$$\mathcal{U}_{ROM} = \{\mathbf{u}\in\mathcal{U}| \quad \mathbf{u}-\mathbf{u}_{ch}\in\mathcal{V}_{ROM}\} \tag{21}$$

The snapshot POD was proposed by Sirovich [20]. The vector $\boldsymbol{\psi}_k$ of the reduced basis are found inside the subspace spanned by the computed displacement fields such that:

$$\boldsymbol{\psi}_k(\mathbf{X}) = \sum_{i=1}^{i=m} \mathbf{U}_i(\mathbf{X}) b_{ik} \quad \forall \mathbf{X} \in \Omega^0 \tag{22}$$

where b_{ik} must maximize the projection λ_k of $\boldsymbol{\psi}_k$ on all the snapshots, such that:

$$\lambda_k = \frac{\sum_{j=1}^{j=m} \left(\int_{\Omega^0} \mathbf{U}_j \cdot \boldsymbol{\psi}_k \, d\Omega^0 \right)^2}{\int_{\Omega^0} \|\boldsymbol{\psi}_k\|^2 \, d\Omega^0} \tag{23}$$

Sirovich proved in [20] that the matrix [**b**] contains the eigenvectors of the covariance matrix [**M**] such that:

$$M_{ij} = \int_{\Omega^0} \mathbf{U}_i \cdot \mathbf{U}_j \, d\Omega^0 \tag{24}$$

[**M**] being a symmetric and positive matrix of size m, the matrix [**b**] contains m eigenvectors. The kth column of [**b**] is related to the eigenvalue λ_k such that $\lambda_k > \lambda_{k+1} \geqslant 0$. Some of the eigenvectors can have a negligible contribution to the approximation Eq. (22). In practice, we remove these negligible contributions by a selection of the $\hat{m}$ first columns of [**b**] such that:

$$\lambda_{\hat{m}} \geqslant \epsilon \lambda_1 \quad \text{and} \quad \lambda_{\hat{m}+1} < \epsilon \lambda_1 \tag{25}$$

where ϵ is a positive parameter of the snapshot POD. In practice we choose $\epsilon = 10^{-8}$. $\mathcal{V}_{ROM}$ being a subspace of $\mathcal{V}_h$, we can introduce a reduction matrix [**A**] such that :

$$\boldsymbol{\psi}_k(\mathbf{X}) = \sum_{j=1}^{j=n} \mathbf{N}_j(\mathbf{X}) A_{jk} \quad \forall \mathbf{X} \in \Omega^0 \tag{26}$$

The governing equations of the reduced-order model can be simply obtained by a Galerkin formulation replacing $\mathcal{U}$ by $\mathcal{U}_{ROM}$ and $\mathcal{V}$ by $\mathcal{V}_{ROM}$ in the continuous formulation (7). The reduced-order problem to solve is: find $\mathbf{u} \in \mathcal{U}_{ROM}$ defined by the constitutive equations and the principle of virtual work:

$$\int_{\Omega^0} \boldsymbol{\epsilon}(\mathbf{u}^*, \mathbf{u}) : \boldsymbol{\Sigma}(\mathbf{F}(\mathbf{u}), \tau \leqslant t) \, d\Omega^0 - \int_{\partial_f \Omega^0} \mathbf{u}^* \cdot \mathbf{f}(\mathbf{X}, t; \{\mathbf{p}\}_\alpha) \, d\Gamma^0 = 0 \quad \forall \mathbf{u}^* \in \mathcal{V}_{ROM} \tag{27}$$

This formulation has no impact on the way the internal variables are computed. As it is done for the finite element method, the stress and the internal variables are computed at Gauss points of the mesh depending on the tensor gradient **F**, using a classical integration scheme. A classical Newton–Raphson algorithm enables to find a displacement field that fulfill the above equilibrium condition.

The drawback of the snapshot POD method is the lack of criteria to choose both convenient snapshots and convenient preliminary simulations (i.e. convenient parameter values) to create snapshots. The quality of a POD approximation depends on the extent of the subspace span by the snapshots. If the POD approximation is not accurate enough new snapshots have to be added into the series of known displacements to extend the POD basis.

The quality of the ROM prediction can be checked using the residual {**R**} of the finite element equilibrium condition such that:

$$R_j(\{\mathbf{q}\}, t, \{\mathbf{p}\}_\alpha) = \int_{\Omega^0} \boldsymbol{\epsilon}(\mathbf{N}_j, \mathbf{u}) : \boldsymbol{\Sigma}(\mathbf{F}(\mathbf{u}_\tau), \tau \leqslant t) \, d\Omega^0 - \int_{\partial_f \Omega^0} \mathbf{N}_j \cdot \mathbf{f}(\mathbf{X}, t) \, d\Gamma^0 \tag{28}$$

with

$$\mathbf{u}(\mathbf{X}, t, \{\mathbf{p}\}_\alpha) = \sum_{i=1}^{i=n} \mathbf{N}_i(\mathbf{X}) q_i(t, \{\mathbf{p}\}_\alpha) \tag{29}$$

A relative error can be defined as follow:

$$\eta = \max_{t \in [0,T]} \frac{\|\{\mathbf{R}\}([\mathbf{A}] \cdot \{\mathbf{a}\}, t, \{\mathbf{p}\}_\alpha)\|}{\|\{\mathbf{R}\}(0, t, \{\mathbf{p}\}_\alpha)\|} \tag{30}$$

$\boldsymbol{\epsilon}$being a linear function of $\mathbf{u}^*$, the matrix form of the reduced equilibrium Eq. (27) is the following:

$$[\mathbf{A}]^T \cdot \{\mathbf{R}\}([\mathbf{A}] \cdot \{\mathbf{a}\}, t, \{\mathbf{p}\}_\alpha) = 0 \tag{31}$$

3. The adaptative incremental algorithm

The multi-level APHR method provides state estimations by summing finite element corrections to ROM predictions. The reduced-state variables of the ROM being global and the FE variables being nodal unknowns, the proposed approximation is multi-level. If at the end of a time increment, the ROM prediction is accurate enough, no finite element correction is performed. On the contrary, the results of the FE correction enable to adapt the ROM. When a FE correction has been performed an approximate state evolution is known. Therefore we can apply the adaptive algorithm proposed in [34]. The mechanical state is taken into account to expand the subspace spanned by the reduced-basis related to the ROM. To master the growth of the ROM, the adaptive procedure involves a POD of the reduced-state variables.

The multi-level approximation is the following:

$$\mathbf{u}(\mathbf{X}, t, \{\mathbf{p}\}_\alpha) = \mathbf{u}^{(n)}_{ROM}(\mathbf{X}, t, \{\mathbf{p}\}_\alpha) + \delta\mathbf{u}^{(n)}_h(\mathbf{X}, t, \{\mathbf{p}\}_\alpha) \quad \mathbf{u}^{(n)}_{ROM} \in \mathcal{V}^{(n)}_{ROM} \quad \delta\mathbf{u}^{(n)}_h \in \mathcal{V}_h \tag{32}$$

The index (n) is the version of the reduced approximation. Such a multi-level approximation enables a straightforward introduction of the adaptive procedure. If $\delta\mathbf{u}^{(n)}_h$ and the shape functions $\boldsymbol{\psi}^{(n)}_k(\mathbf{X})$ of the ROM are not linearly dependent, an extended ROM is obtained such that $\mathcal{V}^{(n+1)}_{ROM} = \text{span}\left\{\boldsymbol{\psi}^{(n)}_1, \ldots, \boldsymbol{\psi}^{(n)}_m, \delta\mathbf{u}^{(n)}_h\right\}$. This last ROM provides a separated representation of the displacement using fields defined over Ω^0 and scalar functions of time and parameters.

Using a numerical one-step time integration scheme, one can forecast different states of the system at different time instants. According to an incremental formulation, the mechanical state is assumed to be known at time instant t_i. The unknowns are the state variables at time t_{i+1}. Several stages are introduced to forecast the mechanical state over the time increment $]t_i, t_{i+1}]$:

- *Stage one*: The reduced-state variables related to the displacement field are forecast assuming $\delta\mathbf{u}^{(n)}_h = 0$.
- *Stage two*: The relative error defined in Eq. (30) estimates the accuracy of the reduced approximation.
- *Stage three*: If the reduced approximation is not accurate enough, then the FE correction $\delta\mathbf{u}^{(n)}_h$ is computed.
- *Stage four*: If the FE correction has been performed, then the reduced-bases are adapted by using the results of the correction stage (stage three).

If the ROM prediction is accurate enough, then no FE computation is performed. In practice, the adaptation of the ROM is achieved only using few FE increments. Moreover, in case of nonlinear equations or parallel computing, the iterative solver takes advantage of the prediction provided by the ROM. Such property of ROM initialization of a nonlinear finite element solution can also be found in [35]. In [9], the author suggests to use a reduced integration domain to reduce the computational time of the reduced-state variables during stage one. As mention in [9,26,34,36], this reduced integration scheme gives rise to additional approximation errors. The aim of this paper being accurate reduced-order simulations of damage problem, we do not introduce this reduced integration scheme here.

The proposed algorithm can be viewed as a two steps Newton–Raphson algorithm. During the stage one the prediction belongs to the functional space related to the ROM and during the stage three the correction belongs to the classical FE functional space.

At the end of stage three, a full state estimation is known. Then the adaptive algorithm proposed in [26] devoted to known state evolutions can be applied. The adaptation stage involves an expansion of the subspaces related to the ROM and a selection of the most significant events using a POD of the reduced-state variables. The extension of the reduced basis is provided by orthogonal contributions extracted from the state corrections. During the adaptation of the ROM the reduced-state variables related to the previous computations are updated. The detailed formulation of the four stages of the algorithm is given below:

Stage one, the unknown are the reduced-state variables $\{\mathbf{a}\}^{(n)}(t_{i+1}, \{\mathbf{p}\}_\alpha)$ and the internal variables over Ω^0 such that:

$$\mathbf{u}(\mathbf{X}, t, \{\mathbf{p}\}_\alpha) = \sum_{k=1}^{k=\bar{m}} \boldsymbol{\psi}_k^{(n)}(\mathbf{X}) a_k^{(n)}(t, \{\mathbf{p}\}_\alpha) + \mathbf{u}_{ch}(\mathbf{X}, t) \tag{33}$$

$$\int_{\Omega^0} \boldsymbol{\epsilon}(\mathbf{u}^*, \mathbf{u}) : \boldsymbol{\Sigma}(\mathbf{F}(\mathbf{u}), \tau \leqslant t)\, d\Omega^0 - \int_{\partial_f \Omega^0} \mathbf{u}^* \cdot \mathbf{f}(\mathbf{X}, t; \{\mathbf{p}\}_\alpha)\, d\Gamma^0 = 0 \quad \forall \mathbf{u}^* \in \mathcal{V}_{ROM}^{(n)} \tag{34}$$

Stage two, evaluation of the error indicator η_{ROM} related to the residue of the finite element equilibrium equations:

$$\eta_{ROM} = \frac{\|\{\mathbf{R}\}\left([\mathbf{A}]^{(n)} \cdot \{\mathbf{a}\}^{(n)}, t, \{\mathbf{p}\}_\alpha\right)\|}{\|\{\mathbf{R}\}(0, t, \{\mathbf{p}\}_\alpha)\|} \tag{35}$$

Stage three, if $\eta_{ROM} < \epsilon_R$ then $\delta\mathbf{u}_h(\mathbf{X}, t_{i+1}, \{\mathbf{p}\}_\alpha) = 0$, else the correction of the displacements is such that:

$$\int_{\Omega^0} \boldsymbol{\epsilon}(\mathbf{u}^*, \mathbf{u}_{ROM}^{(n)} + \delta\mathbf{u}_h^{(n)}) : \boldsymbol{\Sigma}(\mathbf{F}(\mathbf{u}_{ROM}^{(n)} + \delta\mathbf{u}_h^{(n)}), \tau \leqslant t)\, d\Omega^0 - \int_{\partial_f \Omega^0} \mathbf{u}^* \cdot \mathbf{f}(\mathbf{X}, t; \{\mathbf{p}\}_\alpha)\, d\Gamma^0 = 0 \quad \forall \mathbf{u}^* \in \mathcal{V}_h \tag{36}$$

Stage four, if $\eta_{ROM} \geqslant \epsilon_R$ then the subspace $\mathcal{V}_{ROM}^{(n)}$ can be extended by using $\delta\mathbf{u}_h^{(n)}$. First we extend the basis and then we preform a POD of the state variables of the ROM. The intermediate basis is denoted $(\boldsymbol{\psi}_k^{(n+1/2)})_{k=1\ldots s+1}$. We only consider the correction related to the orthogonal projection of $\delta\mathbf{u}_h^{(n)}$ into $\mathcal{V}_{ROM}^{(n)}$. This correction is denoted $\delta_\perp\mathbf{u}_h^{(n)}$ and it is such that:

$$\delta_\perp\mathbf{u}_h^{(n)}(x, t_{i+1}, \{\mathbf{p}\}_\alpha) = \delta\mathbf{u}_h^{(n)}(\mathbf{X}, t_{i+1}, \{\mathbf{p}\}_\alpha) - \widehat{\delta\mathbf{u}_h}^{(n)}(\mathbf{X}, t_{i+1}, \{\mathbf{p}\}_\alpha) \tag{37}$$

$$\widehat{\delta\mathbf{u}_h}^{(n)}(\mathbf{X}, t_{i+1}, \{\mathbf{p}\}_\alpha) = \sum_{k=1}^{k=s} \boldsymbol{\psi}_k^{(n)}(\mathbf{X}) \delta a_k^{(n)}(t_{i+1}, \{\mathbf{p}\}_\alpha) \tag{38}$$

$$\{\delta\mathbf{a}\}^{(n)}(t_{i+1}, \{\mathbf{p}\}_\alpha) = \arg\min_{\{\mathbf{y}\}} \int_{\Omega(\{\mathbf{p}\}_\alpha)} \left\| \delta\mathbf{u}_h(\mathbf{X}, t_{i+1}, \{\mathbf{p}\}_\alpha) - \sum_{k=1}^{k=s} \boldsymbol{\psi}_k^{(n)}(\mathbf{X}) y_k \right\|^2 d\Omega \tag{39}$$

Then, an extended basis $(\boldsymbol{\psi}_k^{(n+1/2)})_{k=1\ldots s+1}$ is built preserving the previous predictions such that:

$$\boldsymbol{\psi}_k^{(n+1/2)} = \boldsymbol{\psi}_k^{(n)} \quad k \leqslant s \tag{40}$$

$$\boldsymbol{\psi}_{s+1}^{(n+1/2)} = \frac{1}{\|\delta_\perp\mathbf{u}_h\|} \delta_\perp\mathbf{u}_h \tag{41}$$

$$a_k^{(n+1/2)}(\tau, \{\mathbf{p}\}_\alpha) = a_k^{(n)}(\tau, \{\mathbf{p}\}_\alpha) \quad \tau \leqslant t_{i+1} \quad k \leqslant s \tag{42}$$

$$a_k^{(n+1/2)}(\tau, \{\mathbf{p}\}_\beta) = a_k^{(n)}(\tau, \{\mathbf{p}\}_\beta) \quad \tau \leqslant T \quad k \leqslant s \quad \beta < \alpha \tag{43}$$

$$a_{s+1}^{(n+1/2)}(\tau, \{\mathbf{p}\}_\beta) = 0 \quad \tau \leqslant T \quad \beta < \alpha \tag{44}$$

$$a_{s+1}^{(n+1/2)}(\tau, \{\mathbf{p}\}_\alpha) = 0 \quad \tau \leqslant t_i \tag{45}$$

$$a_{s+1}^{(n+1/2)}(t_{i+1}, \{\mathbf{p}\}_\alpha) = \|\delta_\perp\mathbf{u}_h\| \tag{46}$$

A POD decomposition of the reduced-variables is performed to avoid a constant growth of the size of the ROM. This POD consists in finding the vectors $(\{\mathbf{V}\}_l)_{l=1\ldots s+1}$ maximizing the following projection on the forecast reduced-state variables:

$$\lambda_l^{(n+1)} = \frac{\sum_{\beta=1}^{\beta=\alpha-1} \int_0^T \left(\{\mathbf{a}\}^{(n+1/2)T}(t, \{\mathbf{p}\}_\beta) \cdot \{\mathbf{V}\}_l\right)^2 dt + \int_0^{t_{j+1}} \left(\{\mathbf{a}\}^{(n+1/2)T}(t, \{\mathbf{p}\}_\alpha) \cdot \{\mathbf{V}\}_l\right)^2 dt}{\|\{\mathbf{V}\}_l\|^2} \tag{47}$$

within the conditions $\|\{\mathbf{V}\}_l\| = 1$ and $\lambda_l \geqslant \lambda_{l+1}$. The vectors $(\{\mathbf{V}\}_l)_{l=1\ldots s+1}$ are the eigenvectors of the covariance matrix $[\mathbf{C}]^{(n+1/2)}$ such that:

$$[\mathbf{C}]^{(n+1/2)} = \sum_{\beta=1}^{\beta=\alpha-1} \int_0^T \{\mathbf{a}\}^{(n+1/2)}(t, \{\mathbf{p}\}_\beta) \cdot \{\mathbf{a}\}^{(n+1/2)T}(t, \{\mathbf{p}\}_\beta)\, dt + \int_0^{t_{j+1}} \{\mathbf{a}\}^{(n+1/2)}(t, \{\mathbf{p}\}_\alpha) \cdot \{\mathbf{a}\}^{(n+1/2)T}(t, \{\mathbf{p}\}_\alpha)\, dt \tag{48}$$

The covariance matrix takes into account the previous results to preserve the ability of the ROM to model the related mechanical states. Then, the main events are selected using the following criteria:

$$[\mathbf{V}] = [\{\mathbf{V}\}_1, \ldots, \{\mathbf{V}\}_{\tilde{s}}] \quad \text{with} \quad \lambda_l > \epsilon_{POD}\lambda_1 \tag{49}$$

$\tilde{s}$ is the new size of the ROM. At last, the new ROM is a POD reduction of the previous ROM:

$$\boldsymbol{\psi}_l^{(n+1)} = \sum_{k=1}^{k=s+1} \boldsymbol{\psi}_k^{(n+1/2)} V_{kl} \quad \forall l = 1 \ldots \tilde{s} \tag{50}$$

$$a_l^{(n+1)}(t, \{\mathbf{p}\}_\beta) = \sum_{k=1}^{k=s+1} a_k^{(n+1/2)}(t, \{\mathbf{p}\}_\beta) V_{kl} \quad \forall l = 1 \ldots \tilde{s} \tag{51}$$

At the end of each simulation, a last POD of the reduced-state variables is performed. The parameters of the method are the coefficients ϵ_R and ϵ_{POD}. The initial ROM $((\boldsymbol{\psi}_k^{(0)}, a_k^{(0)})_{k=1\ldots s})$ can be empty ($s = 0$). In such a case, the first increment of the simulation starts with one full FE increment (stage three). The adaptive algorithm proposed in [26] being used, the following property is fulfilled. The decomposition obtained at the end of the incremental computation is the POD of a state prediction having an accuracy related to ϵ_R. The proof of this property can be found in [26]. When during a simulation no correction is performed at stage three, the algorithm provides a classical POD prediction of the mechanical state.

The three following properties hold. If the accuracy of the ROM is good enough, or if ϵ_R is large enough, the multi-level APHR algorithm provides the classical solution related to the ROM without any adaptation. If ϵ_R is small enough, the ROM prediction is corrected at each time increment, therefore the multi-level APHR algorithm provides the FE solution. If the size of the ROM if sufficiently small (ϵ_{POD} sufficiently large), the ROM prediction has no effect on the FE correction.

4. Numerical results

4.1. Investigation of state variations induced by modification of the loading conditions

In this section the results provided by the APHR method are compared with the ones provided by the snapshot POD method, and the FE method. All the numerical results were obtained using the same FE research code ZeBuLon [37]. The mesh and the boundary conditions of the mechanical problem are shown in Fig. 1. This is

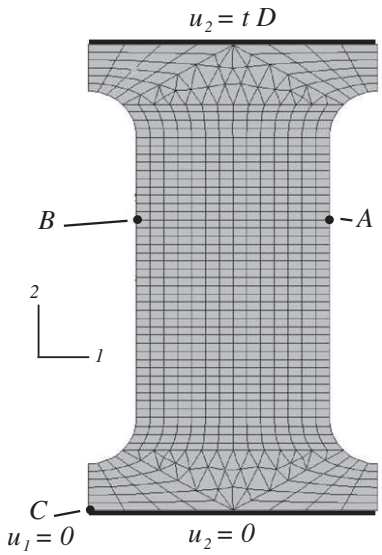


Fig. 1. Mesh and boundary conditions, load perturbations are imposed at node *A* and node *B*.

a 2D plane stress problem. It represents a specimen used for tensile tests. The set of material parameters taken for the Rousselier law are : the Young modulus is 200 GPa, the Poisson's ratio is 0.3, $D = 2.$, $\sigma_1 = 490$ MPa. The hardening law $\sigma_f(p)$ is given in Table 1.

A series of three simulations is considered. The variable parameters of the models are the nodal forces on direction 2 at node *A* and node *B* (Fig. 1). They are respectively the first, and the second variable parameters of the model. The three simulations are related to $\{\mathbf{p}\}_1^T = \{0, 0\}$, $\{\mathbf{p}\}_2^T = \{10\text{N}, 0\}$, and $\{\mathbf{p}\}_3^T = \{0, 10\text{N}\}$. For the two last cases, the load perturbation is 0.03% of the maximal force applied on the specimen. A first strain localization appears around time instant $t = 4$ s. To perform robust FE simulations the time step has been set to 0.002 s in time interval [0, 4 s] and 0.0002 s in the time interval [4 s, 5.5 s]. The force–time curves related to the last three FE simulations are shown in Fig. 2.

Table 1
Hardening law.

σ_f (MPa)	p
495	0
530	0.00732323
570	0.0171212
604	0.0269495
634	0.036798
660	0.0466667
681	0.0565606
700	0.0664646
716	0.0763838
730	0.0863131
740	0.0962626
749	0.106217
755	0.116187
765	0.126136
770	0.133111
771	0.136106
778	0.146071
809	0.195914
855	0.295682
889	0.39551
917	0.495369
940	0.595253
960	0.695152
977	0.795066
993	0.894985
1008	0.994909

The load perturbation at node *A* has no significant effect on the finite element solution. The finite element simulations related to parameter values $\{\mathbf{p}\}_1$ and $\{\mathbf{p}\}_2$ are very similar. But the last value of parameters has a huge influence on the solution when the bifurcation starts.

The continuous model having multiple solutions, all of the Finite element ones are acceptable approximate solutions. But, each finite element simulation aims to forecast one solution. The adaptive reduced-order modeling method must be able to forecast a convenient approximate solution close to the solution of the finite element problem being reduced. In other word, the simulations using adaptive reduced approximation must be guided by given perturbations like the FE simulations.

4.2. Investigation of state variations induced by reduced-order approximations

The POD basis being created using the results of the first FE simulation, it does not make sense to restart this simulation using this POD basis, because the FE solution is necessarily known. Only simulations related to $\{\mathbf{p}\}_2$ and $\{\mathbf{p}\}_3$ have been restarted using the POD approximation. But the three simulations have been restarted using the APHR method with an empty initial basis. The parameter ϵ_{POD} has been set to 10^{-8}. The value of ϵ_R has been set to 0.005. Therefore we investigate the capability of the reduced approximations to estimate the FE solutions without knowing it.

The results provided by the finite element method and the APHR method are very close. The error on cumulated plastic strain is less than 5%. The comparison of the force–time curve and the cumulated plastic strain at the end of the time interval of the first simulation are shown in Fig. 3 for the first simulation. Three-hundred and nine finite element corrections has been performed to build the adapted reduced approximation during the first APHR simulation ($n = 309$). An example of such a correction at time $t = 0.4084s$ is shown in Fig. 4. At the end of this simulation the dimension of subspace $\mathcal{V}_{ROM}^{(309)}$ is 10. The number of global linear solutions was 3804 for the finite element simulation and 493 for the APHR simulation. Therefore the number of global linear solution has been divided by a factor 7.7. Sequences of linear FE solutions during the first simulation are shown in Fig. 5 for both FE and APHR simulations. One can see that correction of the ROM approximation is performed only during few time increments. Moreover, when a correction is computed the Newton–Raphson algorithm takes advantage of the ROM prediction. In case of FE simulation, one can notice that the convergence of the residues is not monotonous. The second and the third simulations are performed with similar results for both FE method and APHR method respectively.

Because the first variable parameter has low influence on the FE solution, the POD basis created to represent the results of the first simulation is very efficient to forecast the FE solution of the second simulation. The error on cumulated plastic strain was lower than 1%. But the POD basis failed to forecast accurately the FE solution of the third simulation. It provided the solution of the first simulation as solution of the third simulation. Despite it is one of the possible solutions, it is not the expected solution of the third simulation. Therefore it is not convenient. The comparison of the force–time curve and the cumulated plastic strain at the end of the time interval of the third simulation are shown in Fig. 6.

During the third simulation, after bifurcation of the solution, the error due to the POD approximation is growing up. It is not the case of the error related to the adaptive reduced approximation built using the APHR method. This is illustrated by the evolution of

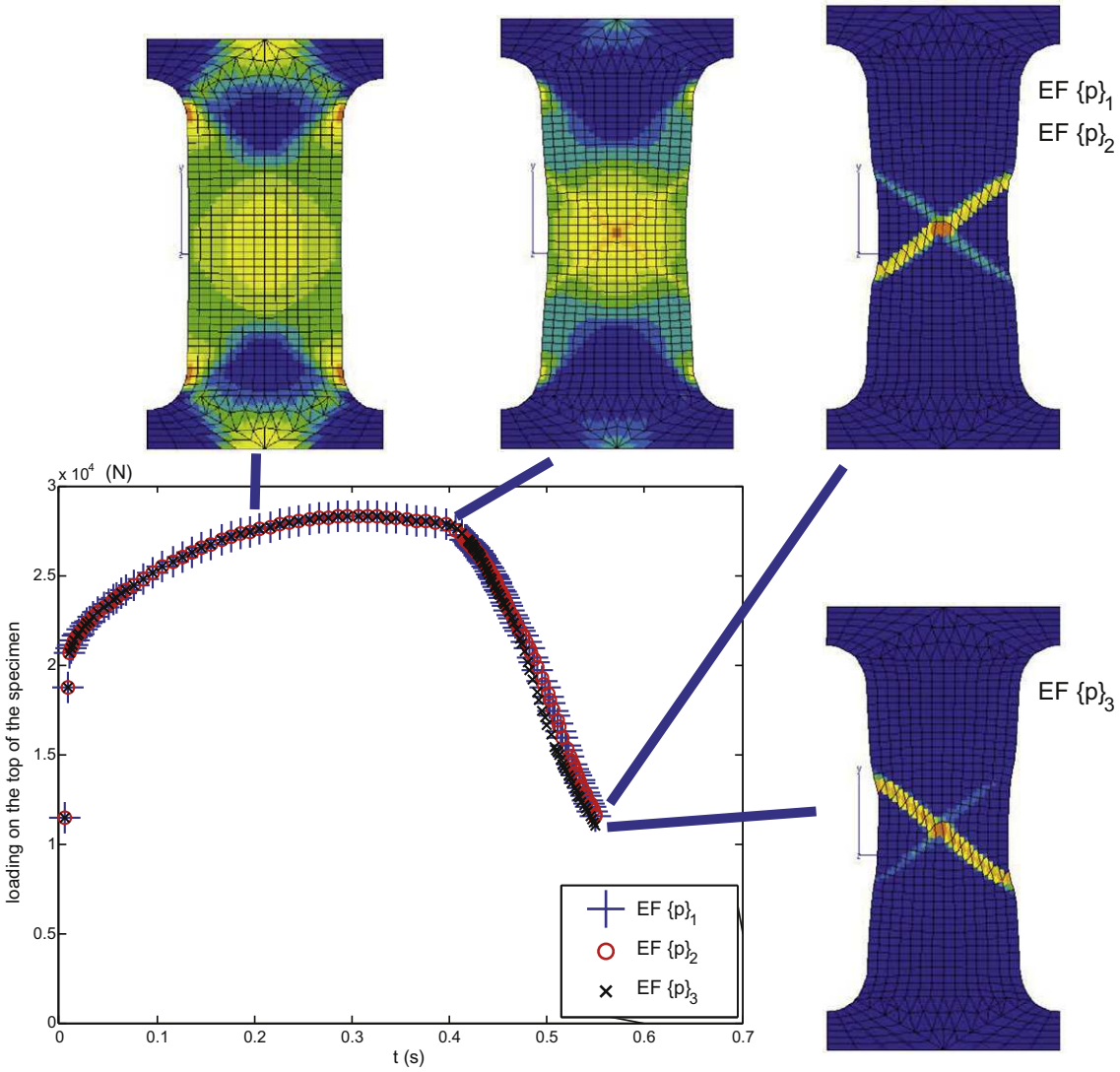


Fig. 2. Effects of the load perturbations on the force–time curve and on the cumulated plastic strain.

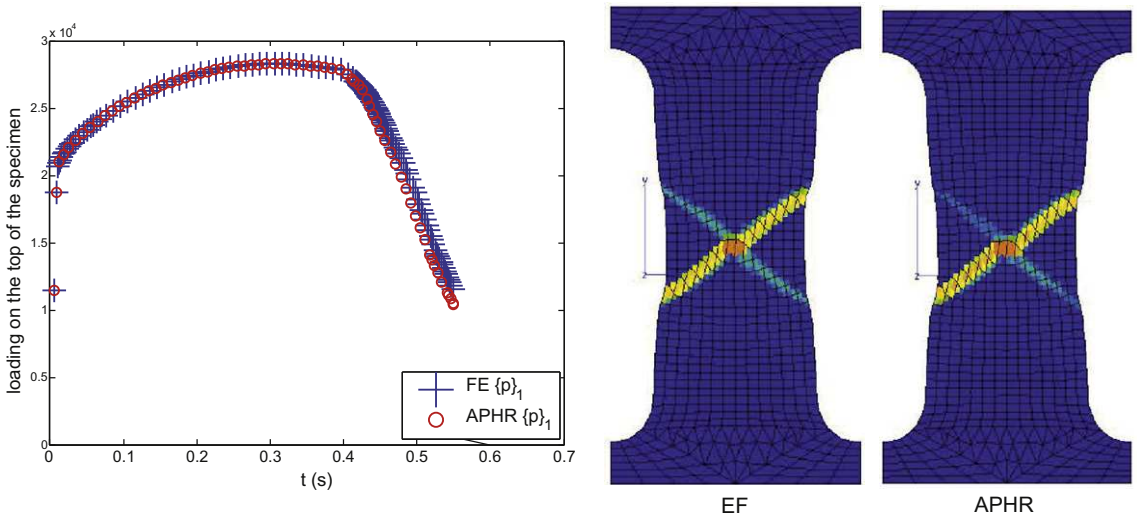


Fig. 3. Force–time curve and the cumulated plastic strain at the end of the time interval related to $\{\mathbf{p}\}_1$.

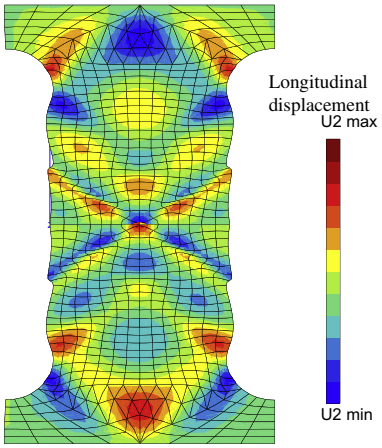


Fig. 4. Example of field $\delta \mathbf{u}_h^{(\eta)}(\mathbf{X}, t, \{\mathbf{p}\}_1)$ added to the reduced-basis at time $t = 0.4084$ s.

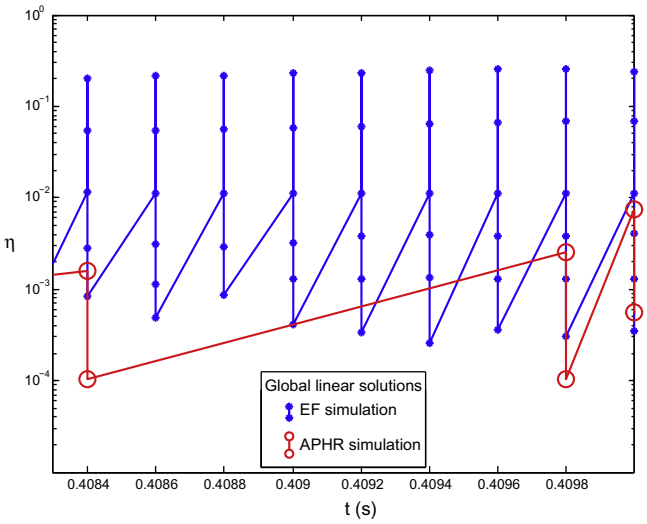


Fig. 5. Each marker corresponds to a linear solution of FE equilibrium equations.

the cumulative plasticity at a Gauss point close to node B shown in Fig. 7.

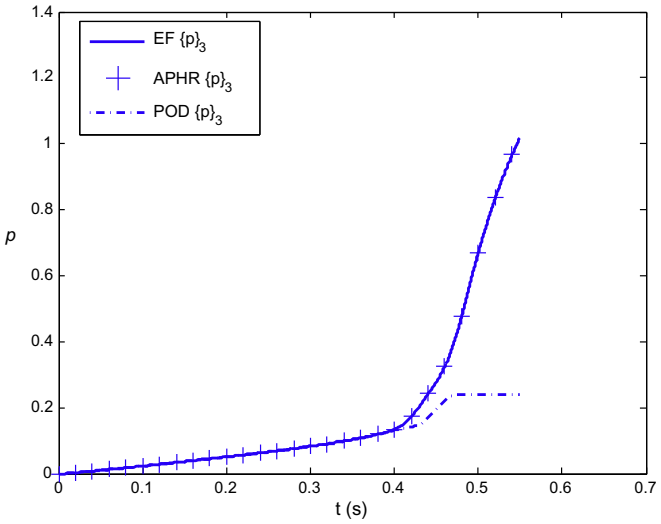


Fig. 7. Cumulated plastic strain on a Gauss point close to node B.

5. Conclusion

Previous works on the APHR method have been pursued to obtain a multi-level APHR method. The quality of the ROM prediction is controlled using the residual of the FE equilibrium equations and a FE correction. If needed the ROM basis is expanded using the FE corrections. These corrections can be viewed as new snapshots efficiently selected. This algorithm provides an adapted POD basis that represents all the previous numerical results obtained during the simulation. Therefore it provides a separated representation of the forecast displacements. By setting the two parameters of the adaptive algorithm, ϵ_{POD} and ϵ_R, one can obtain a full FE simulation, or are a full ROM simulation, or an intermediate scheme mixing FE and ROM predictions.

Despite the solution of damage problem is not unique in case of bifurcation, the multi-level APHR method combined with a robust integration scheme enables to forecast one mechanical state per simulation. Therefore the adaptive reduced simulations can be guided toward particular unknown solution like robust FE simulations. If the reduced approximation is not adapted in case of bifurcation, it is not possible to guide the solution introducing perturbations in the model. The proposed method have proved its efficiency on Rousselier constitutive equations. The number of global linear solutions needed to estimate the FE solution has been

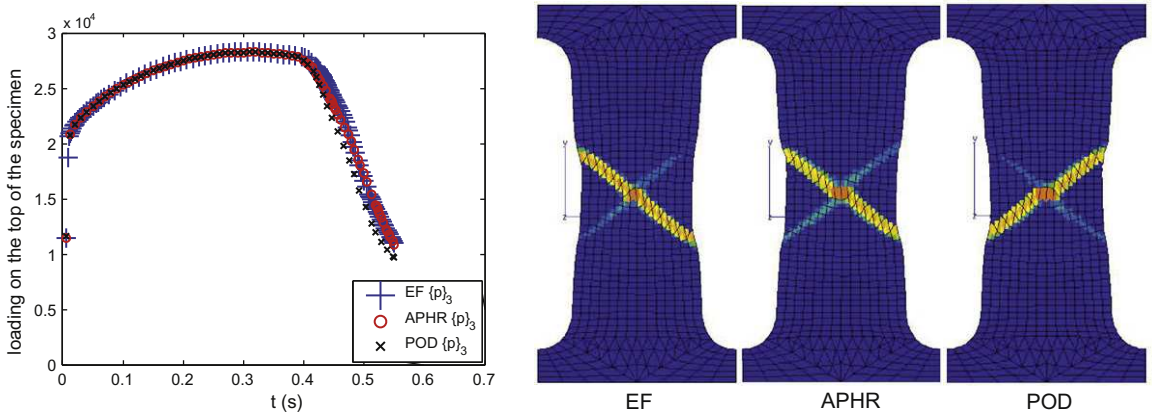


Fig. 6. Effects of the load perturbation imposed at node *B* (force–displacement curve and cumulated plastic strain).

divided by 7.7. The error on cumulated plastic strain was always lower than 5%.

The case of large number of simulations involved in the series of problems needs deeper research. The extent of the efficiency of the multi-level APHR method remains an open question.

Acknowledgment

This work has been funded by EDF.

References

[1] J. Lemaitre, A. Benallal, D. Marquis, Nuclear Engineering and Design 133 (1992) 345–360.
[2] B. Tanguy, J. Besson, R. Piques, A. Pineau, Engineering Fracture Mechanics 72 (2005) 413–434.
[3] M. Mazière, J. Besson, S. Forest, B. Tanguy, H. Chalons, F. Vogel, European Journal of Mechanics A/Solids 28 (2009) 36–44.
[4] G. Pijaudier-Cabot, A. Benallal, International Journal of Solids and Structures 30 (1993) 1761–1775.
[5] A. Benallal, Archives of Mechanics 44 (1) (1992) 15–29.
[6] G. Barbier, A. Benallal, V. Cano, Comptes Rendus de l'Académie des Sciences, Paris 326 (1998) 153–158.
[7] G. Barbier, G. Rousselier, R. Lebrun, Y. Sun, Le Journal de Physique IV 8 (1999) 19–27.
[8] J. Besson, D. Steglich, W. Brocks, International Journal of Solids Structures 38 (2001) 8259–8284.
[9] D. Ryckelynck, International Journal of Computational Physics 202 (2005) 346–366.
[10] G. Rousselier, Finite deformation constitutive relations including ductile fracture damage, in: S. Nemat-Nasser (Ed.), Three-dimensional Constitutive Relations and Ductile Fracture, North-Holland, Amsterdam, 1981, pp. 331–355.
[11] G. Rousselier, Nuclear Engineering and Design 105 (1987) 97–111.
[12] D.N. Daescu, I.M. Navon, International Journal for Numerical Methods in Fluids 53 (6) (2007) 985–1004.
[13] S. Ganapathysubramanian, N. Zabaras, Computer Methods in Applied Mechanics and Engineering 193 (2004) 5017–5034.
[14] E. Monteiro, J. Yvonnet, Q.C. He, Computational Materials Science 42 (2008) 704–712.
[15] O. Balima, Y. Favennec, M. Girault, D. Petit, International Journal for Numerical Methods in Engineering 67 (7) (2006) 895–915.
[16] S. Niroormandi, I. Alfaro, E. Cueto, F. Chinesta, Computer Methods and Programs in Biomedicine 91 (2008) 223–231.
[17] K. Karhunen, Über lineare Methoden in der Wahrscheinlichkeitsrechnung, Ann. Acad. Sci. Fenn., Ser. A1: Math.-Phys. 37 (1947) 3–79.
[18] M.M. Loéve, Probability Theory, The University Series in Higher Mathematics, third ed., Van Nostrand, Princeton, NJ, 1963.
[19] P. Holmes, J.-L. Lumley, B.G. Turbulence, Coherent Structures, Dynamical Systems and Symmetry, first ed., Cambridge University Press, 1998.
[20] L. Sirovich, Quarterly of Applied Mathematics XLV (3) (1987) 561–571.
[21] P. Ladevèze, Comptes Rendus de l'Académie des Sciences, Paris Sèrie II 300 (2) (1985) 41–44.
[22] O. Allix, P. Vidal, Computer Methods in Applied Mechanics and Engineering 191 (2002) 2727–2758.
[23] P.A. Boucard, S. Buytet, P.A. Guidault, A multiscale strategy for structural optimization, International Journal for Numerical Methods in Engineering 78 (1) (2009) 101–126.
[24] P. Ladevèze, A. Nouy, Computer Methods in Applied Mechanics and Engineering 192 (2003) 3061–3087.
[25] D. Ryckelynck, Comptes Rendus Mécanique 330 (2002) 499–505.
[26] D. Ryckelynck, F. Chinesta, E. Cueto, A. Ammar, State of the Art Reviews 13 (2006) 91–128.
[27] J. Besson, D. Steglich, W. Brocks, International Journal of Plasticity 19 (2003) 1517–1541.
[28] H. Ziegler, Some extremum principles in irreversible thermodynamics with applications to continuum mechanics, in: I.N. Sneddon, R. Hill (Eds.), Progress in Solid Mechanics, vol. IV, North-Holland, Amsterdam, 1963.
[29] P. Germain, Q.S. Nguyen, P. Suquet, Journal of Applied Mechanics 50 (1983) 1010–1020.
[30] J. Lemaitre, J.-L. Chaboche, Mecanique des materiaux solides, first ed., Dunod, Paris, 1985 (English version published by Cambridge University Press, Cambridge, 1990).
[31] F. Sidoroff, A. Dogui, International Journal of Solids Structures 38 (2001) 9569–9578.
[32] E. Lorentz, J. Besson, V. Cano, Computer Methods in Applied Mechanics and Engineering 197 (2008) 1965–1982.
[33] O. Zienkiewicz, R.L. Taylor, Finite Element Method, vol. 1–3, Butterworth-Heinomann, London, 2000.
[34] D. Ryckelynck, D.-M. Benziane, Computer Methods in Applied Mechanics and Engineering 199 (17-20) (2010) 1134–1142.
[35] R. Markovinovic, J.D. Jansen, International Journal for Numerical Methods in Engineering 68 (2006) 525–541.
[36] D. Ryckelynck, International Journal for Numerical Methods in Engineering 77 (1) (2009) 75–89.
[37] J. Besson, R. Foerch, Computer Methods in Applied Mechanics and Engineering 142 (1997) 165–187.

Annexe 4

Arch Comput Methods Eng
DOI 10.1007/s11831-010-9055-0

ORIGINAL PAPER

Truncated Integration for Simultaneous Simulation of Sintering Using a Separated Representation

B. Sarbandi · S. Cartel · J. Besson · D. Ryckelynck

Received: 15 March 2010 / Accepted: 15 March 2010

Abstract Recent developments of multidimensional solvers using separated representation make it possible to account for the multidimensionality of mechanical models in materials science when doing numerical simulations. This paper aims to extend the separated representation to inseparable equations using an efficient integration scheme. It focuses on the dependence of constitutive equations on material coefficients. Although these coefficients can be optimized using few experimental results, they are not very well known because of the natural variability of material properties. Therefore, the mechanical state can be viewed as a function depending not only on time and space variables but also on material coefficients. This is illustrated in this paper by a sensitivity analysis of the response of a sintering model with respect to variations of material coefficients. The considered variations are defined around an optimized value of coefficients adjusted by experimental results. The proposed method is an incremental method using an extension of the integration scheme developed for the Hyper Reduction method. During the incremental solution, before the adaptation of the representation, an assumed separation representation is used as a reduced-order model. We claim that a truncated integration scheme enables to forecast the reduced-state variables related to the assumed separated representation. The fact that the integrals involved in the formulation can not be written as a sum of products of one-dimensional integrals, this approach reduces the extent of the integration domain.

B. Sarbandi · S. Cartel · J. Besson · D. Ryckelynck (✉)
Centre des Materiaux, Mines ParisTech, CNRS UMR 7633,
BP 87, 91003 Evry, France
e-mail: david.ryckelynck@ensmp.fr

1 Introduction

Sintering is a key part for producing materials in a useful and robust form, especially ceramics and generally porous materials, and involves heating the porous structure or a powder compact. During this thermal treatment cycle, the part tends to increase in density as sintering proceeds and this still further improves the mechanical properties. Density increasing implies, of course, an overall shrinkage which leads to complex deformations. The shrinkage and deformation of ceramic body during this firing process can be simulated by sintering deformation constitutive model [1]. The thermal and mechanical parameters of the constitutive model should be determined experimentally and introduced in corresponding finite element analysis. It is obvious that whereas the results of finite element simulation of sintering deformation will be significantly affected by variations in some of these parameter values, the effect of similar variations in other parameter values will be of little consequences. Consequently, investigation of the effect of parameter variations are important in so far as they provide a guide to the level of uncertainty of the parameters. In other words, the parameters that have the smallest effect could be identified for a large class of materials. The parameters having a greater effect should be specified with greater accuracy for each considered materials. We propose to quantify how sensitive the results of a finite element analysis were to disturbance of the mechanical parameters of the sintering constitutive model. Therefore, the mechanical state can be viewed as a function depending not only on time and space variables but also on material coefficients, in the framework of multidimensional modeling.

Multidimensional modeling of materials seems to be an appealing approach thanks to the recent works of F. Chinesta, A. Ammar and co-authors [2, 3] on separated-

Published online: 06 October 2010

representation-based algorithms. This kind of representation was applied in numerous contexts: (i) quantum chemistry [4]; (ii) Brownian dynamics [5]; (iii) kinetic theory description of polymers solutions and melts [6]; and kinetic theory descriptions of rods suspensions [7]. In this paper the multidimensionality of the model is related to the variability of the material properties.

By using a separated representation it is assumed that equilibrium equations are not rigorously fulfilled. The purpose of this work does not concern the convergence of such representation. It is shown the ability of the proposed method to efficiently estimate the mechanical variables using a separated representation and a truncated integration scheme. For this reason the residue of the equilibrium equations is introduced in the following formulations. The goal of numerical simulation is to forecast a mechanical state related to residue small enough or as small as possible. Considering inseparable equation the method is quite generic. The second section is about the generic formulation of the mechanical problem. The third section explains the truncated integration scheme involved in the separated-representation-based algorithm. The fourth section gives the details of the constitutive model of sintering and the related parameters. The fifth section presents the numerical results and a comparison to a classical FEM. And finally the conclusion is discussed in the sixth section.

2 Formulation of the Continuous Model Using Separated Representation

A generic form of continuous mechanical model is considered, which is a parameterized nonlinear model. $\{p\}$ denotes the column of the model parameters, which are material coefficients. The purpose of the proposed method is to estimate the mechanical state as a function defined in the multidimensional space $\Omega^0 \times \mathcal{P} \times]t_0, t_f]$, where Ω^0 is the reference configuration of the mechanical system, $\mathcal{P}$ is the manifold related to the model parameters and $]t_0, t_f]$ is the time interval. The continuous model is described using the finite strain formalism. The reference configuration Ω^0 can be the domain either at time $t = 0$ (total Lagrangian formulation) or at time t (updated Lagrangian formulation). In order to facilitate the implementation of the method into an existing code, we preserve an incremental scheme. At time instant t, the continuous medium is occupying a domain Ω. The approximated displacement field at time t is defined on $\Omega^0 \times \mathcal{P}$ and it is denoted by $\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, \{p\}, t)$. $\underline{\boldsymbol{X}}$ is denoting the initial position of a material point in Ω^0. It is represented using the following separated representation:

$$\underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, \{p\}, t) = \sum_{j=1}^{j=\gamma} \underline{\boldsymbol{\phi}}_j(\underline{\boldsymbol{X}}) g_j(\{p\}) a_j(t) + \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}, t)$$
$$\forall \underline{\boldsymbol{X}} \in \Omega^0 \; \forall \{p\} \in \mathcal{P} \; \forall t \in]0, T] \tag{1}$$

where $\underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}, t)$ is a given function such that Dirichlet conditions defined on $\partial_U \Omega^0 \times \mathcal{P}$ are fulfilled, $\underline{\boldsymbol{\phi}}_j(\underline{\boldsymbol{X}})$ is a displacement field defined in $H^1(\Omega^0)$, $g_j(\{p\})$ is a scalar function defined in $L^2(\mathcal{P})$, and $a_j(t)$ is a scalar continuous time function. $\underline{\boldsymbol{u}}_c$ is a continuous function of the variable $\underline{\boldsymbol{X}}$ equal to zero almost everywhere in Ω^0 such that $\underline{\boldsymbol{u}}_c$ is equal to the given Dirichlet conditions at any point of $\partial_U \Omega^0 \times \mathcal{P}$. Therefore $\underline{\boldsymbol{\phi}}_j(\underline{\boldsymbol{X}}) = 0$ over $\partial_U \Omega^0$. The displacement field belongs to an affine function space $\mathcal{U}$ defined by:

$$\mathcal{U} = \left\{ \begin{array}{l} \underline{\boldsymbol{u}}(.,.,t) \in H^1(\Omega^0) \otimes L^2(\mathcal{P}) | \\ \exists (\underline{\boldsymbol{\phi}}_j)_{j=1,\ldots,\gamma}, \exists (g_j)_{j=1,\ldots,\gamma}, \exists (a_j)_{j=1,\ldots,\gamma}, \\ \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}},.,t) = \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}, t) \; \forall \underline{\boldsymbol{X}} \in \partial_U \Omega^0 \\ \underline{\boldsymbol{u}}(\underline{\boldsymbol{X}}, \{p\}, t) = \sum_{j=1}^{j=\gamma} \underline{\boldsymbol{\phi}}_j(\underline{\boldsymbol{X}}) g_j(\{p\}) a_j(t) + \underline{\boldsymbol{u}}_c(\underline{\boldsymbol{X}}, t) \end{array} \right\} \tag{2}$$

A vector space denoted $\mathcal{V}$ is introduced such that:

$$\mathcal{V} = \left\{ \underline{\boldsymbol{u}}^*(.,.,t) \in H^1(\Omega^0) \otimes L^2(\mathcal{P}) | \; \underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}},.,t) = 0 \; \forall \underline{\boldsymbol{X}} \in \partial_U \Omega^0 \right\} \tag{3}$$

The second Piola-Kirchhoff stress tensor $\underset{\sim}{\boldsymbol{S}}$ is a nonlinear function of the deformation gradient history depending on the parameters $\{p\}$:

$$\underset{\sim}{\boldsymbol{S}} = \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{F}}_\tau, \tau \leq t; \{p\}) \tag{4}$$

where $\underset{\sim}{\boldsymbol{\Sigma}}$ is a formal operator that must be defined by constitutive equations and $\underset{\sim}{\boldsymbol{F}}_\tau$ is the deformation gradient at time instant τ. Consequently the following general relations are fulfilled:

$$F_{ij} = \delta_{ij} + \frac{\partial u_i}{\partial X_j} \tag{5}$$

$$\underset{\sim}{\boldsymbol{\sigma}} = \frac{1}{\det(\underset{\sim}{\boldsymbol{F}})} \underset{\sim}{\boldsymbol{F}}.\underset{\sim}{\boldsymbol{S}}.\underset{\sim}{\boldsymbol{F}}^T, \tag{6}$$

where $\underset{\sim}{\boldsymbol{\sigma}}$ is the Cauchy stress tensor and δ_{ij} is the Kronecker delta.

The boundary $\partial\Omega^0$ of Ω^0 is denoted by $\partial_U \Omega^0 \cup \partial_f \Omega^0$. On $\partial_f \Omega^0$, there is a given force field $\underline{\boldsymbol{f}}(.,t)$ depending on time t. The statement of the mechanical problem is presented in the following equations whose purpose is to find an

estimation of the displacement field $\underline{\boldsymbol{u}} \in \mathcal{U}$ related to a given degree of precision ϵ_r which is defined by the constitutive equations and the principle of virtual work:

$$\begin{aligned}
&\int_{\Omega^0\times\mathcal{P}} \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*,\underline{\boldsymbol{u}}) : \underset{\sim}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{F}}(\underline{\boldsymbol{u}}), \tau \le t, \{p\})\, d\Omega^0\, dp \\
&\quad - \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{f}}(\underline{\boldsymbol{X}}, t)\, d\Gamma^0\, dp \\
&= \int_{\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_v\, d\Omega^0\, dp \\
&\quad + \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^* . \underline{\boldsymbol{r}}_b\, d\Gamma^0\, dp \quad \forall \underline{\boldsymbol{u}}^* \in \mathcal{V}\ \forall t \in]t_0, t_f] \qquad (7)
\end{aligned}$$

with the following accuracy condition:

$$\begin{aligned}
&\int_{\Omega^0\times\mathcal{P}} \underline{\boldsymbol{r}}_v . \underline{\boldsymbol{r}}_v\, d\Omega^0\, dp \\
&\quad + \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{r}}_b . \underline{\boldsymbol{r}}_b\, d\Gamma^0\, dp \le \epsilon_r \quad \forall t \in]t_0, t_f] \qquad (8)
\end{aligned}$$

where $\underline{\boldsymbol{u}}^*$ is a test function and $\underset{\sim}{\boldsymbol{\varepsilon}}$ is linear function of $\underline{\boldsymbol{u}}^*$ such that:

$$\begin{aligned}
&\varepsilon_{ij}(\underline{\boldsymbol{u}}^*, \underline{\boldsymbol{u}}) \\
&= \frac{1}{2}\left(\frac{\partial u_i^*}{\partial X_j} + \frac{\partial u_j^*}{\partial X_i} + \sum_{k=1}^{k=3} \left(\frac{\partial u_k^*}{\partial X_i}\frac{\partial u_k}{\partial X_j} + \frac{\partial u_k}{\partial X_i}\frac{\partial u_k^*}{\partial X_j} \right) \right) \qquad (9)
\end{aligned}$$

The smaller the residuals $\underline{\boldsymbol{r}}_v$ and $\underline{\boldsymbol{r}}_b$ the better the equilibrium conditions (7) are fulfilled. According to the framework of the irreversible thermodynamic processes, a constitutive law can be defined by a choice of: internal variables $\underset{\sim}{\boldsymbol{z}}$ and a free energy $w(\underset{\sim}{\boldsymbol{\varepsilon}}, \underset{\sim}{\boldsymbol{z}}, \{p\})$ [8]. Some conjugated variables $\underset{\sim}{\boldsymbol{Z}}$ are associated with the internal variables $\underset{\sim}{\boldsymbol{z}}$ using the definition of the dissipation. Examples of elastoplastic or elastoviscoplastic constitutive models can be found in [9, 10]. The detailed equations of the constitutive law related to sintering transformation are given in Sect. 4. It is clear that the existence of the solution of the approximate problem defined by (7) to (9) depends on the value of ϵ_r when the solution of the exact problem can not be represented using a finite sum as proposed in (1). The convenient choice of ϵ_r is an open question which is not addressed in this paper. We assume that ϵ_r is such that a solution of the approximate problem exists.

3 Formulation Using a Truncated Integration Scheme

The solution (1) is not given in advance but is computed adaptively. The key point of an efficient determination of the separated representation is the use of convenient test functions. Usually, the set of equations involved in (4) to (7) are not separable equations because no separated representation of the stress tensor $\underset{\sim}{\boldsymbol{S}}$ can be chosen without introducing new error approximation. Therefore, it is not possible to simplify the computation of the integrals defined over $\Omega^0 \times \mathcal{P}$ or $\partial_f\Omega^0 \times \mathcal{P}$ as a sum of products of integrals defined over $\mathcal{P}$ and Ω^0 or $\partial_f\Omega^0$ as proposed in [3] and [11]. This is the reason why we propose to extend the Hyper Reduction method proposed in [12] and [13] for adaptive reduced-order methods. The simplified integration scheme is coined by the truncated integration method. One can notice that similar truncated integration scheme are also used in Earth science as an approximation of a complete integration scheme [14]. The particularity of the proposed approach is the extension of the truncated integration scheme to boundary value problems.

The adaptive algorithm used to construct the separated representation (1) is a predictor-corrector algorithm. As it is mentioned above, this algorithm is also an incremental algorithm. The time interval is split into time increments. The mechanical state has to be forecast at the end of each time increment knowing the state at the beginning of the time increment. The separated representation is adapted at each time increment, if necessary. The proposed algorithm is very similar to the one proposed in [11] the fact that it is applied step by step over the time interval using a truncated integration scheme. At time instant t, a separated representation is assumed to be known. During the prediction step, the known separated representation of the fields defined over $\Omega^0 \times \mathcal{P}$ represents a reduced-order model. The values of the time functions $(a_j(t))_{j=1..\gamma}$ at the end of the time increment are the reduced state variable of this model. We claim that, if the amount of reduced state variables is small enough, a truncated integration domain can be introduced to formulate the equilibrium equations in order to estimate the reduced state variables. This truncated integration domain is denoted $\mathcal{W}_\Pi$, which is a submanifold of $\Omega^0 \times \mathcal{P}$. The truncated integration domain is the support of truncated test functions related to the known separated representation.

In the formulation of the equilibrium equation proposed above the test functions belong to wide vector space. But various equilibrium conditions can be introduced using different vector spaces provided that the rank of these equations is equal to the number of unknowns. During the prediction step, the number of unknowns is equal to γ, which is the dimension of the reduced-order model defined by the separated representation. Therefore, as proposed in [12] and in [13], we propose to introduce the vector space of truncated test functions denoted $\mathcal{V}_\Pi$ is introduced such that:

$$\begin{aligned}
\mathcal{V}_\Pi = \Big\{ &\underline{\boldsymbol{u}}^*(.,.) \in H^1(\Omega^0) \otimes L^2(\mathcal{P}) | \\
&\underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}}, ., t) = 0\ \forall \underline{\boldsymbol{X}} \in \partial_U\Omega^0
\end{aligned}$$

$$\left.\begin{aligned}
&\underline{\boldsymbol{u}}^*(\underline{\boldsymbol{X}},\{p\},t) = h(\underline{\boldsymbol{X}},\{p\})\sum_{j=1}^{j=\gamma}\underline{\boldsymbol{\phi}}_j(\underline{\boldsymbol{X}})g_j(\{p\})a_j^*\\
&h(\underline{\boldsymbol{X}},\{p\}) \in C^\infty(\Omega^0\times\mathcal{P})\\
&h(\underline{\boldsymbol{X}},\{p\}) = 0\ \forall(\underline{\boldsymbol{X}},\{p\}) \notin \mathcal{W}_\Pi\\
&h(\underline{\boldsymbol{X}},\{p\}) = 1 \text{ almost everywhere over } \mathcal{W}_\Pi
\end{aligned}\right\} \quad (10)$$

Using this vector space for test functions, the integrals over the complementary part of $\mathcal{W}_\Pi$ are equal to zero. Therefore the prediction problem is: knowing $(\underline{\boldsymbol{\phi}}_j)_{j=1,\ldots,\gamma}$ and $(g_j)_{j=1,\ldots,\gamma}$ find $(a_j(t))_{j=1,\ldots,\gamma}$ such that $\underline{\boldsymbol{u}} \in \mathcal{U}$ minimizes η_Π and such that:

$$\begin{aligned}
\text{and } \eta_\Pi = &\int_{\mathcal{W}_\Pi} \underline{\boldsymbol{r}}_v.\underline{\boldsymbol{r}}_v\, d\Omega^0\, dp\\
&+\int_{\partial_f\Omega^0\times\mathcal{P}\cap\mathcal{W}_\Pi} \underline{\boldsymbol{r}}_b.\underline{\boldsymbol{r}}_b\, d\Gamma^0\, dp \qquad (11)\\
\int_{\mathcal{W}_\Pi} &\underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*,\underline{\boldsymbol{u}}) : \underset{\approx}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{F}}(\underline{\boldsymbol{u}}),\tau\le t,\{p\})\, d\Omega^0\, dp\\
&-\int_{\partial_f\Omega^0\times\mathcal{P}\cap\mathcal{W}_\Pi} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{f}}(\underline{\boldsymbol{X}},t)\, d\Gamma^0\, dp\\
=&\int_{\mathcal{W}_\Pi} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_v\, d\Omega^0\, dp\\
&+\int_{\partial_f\Omega^0\times\mathcal{P}\cap\mathcal{W}_\Pi} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{r}}_b\, d\Gamma^0\, dp \quad \forall\underline{\boldsymbol{u}}^* \in \mathcal{V}_\Pi \qquad (12)
\end{aligned}$$

This prediction step is followed by the correction step if $\text{mes}(\Omega^0\times\mathcal{P})\eta_\Pi > \text{mes}(\mathcal{W}_\Pi)\epsilon_r$. The correction step consists of finding an adapted separated representation solving (7). This equation being nonlinear with respect to $\underline{\boldsymbol{u}}$, finding its solution takes advantage of the prediction step. This solution is performed using a Newton-Raphson algorithm, which is an iterative algorithm. Several corrections $\underline{\boldsymbol{\delta u}}$ are added to the prediction $\underline{\boldsymbol{u}}$ solving the following linear problem with respect to $\underline{\boldsymbol{\delta u}}$: find $\underline{\boldsymbol{R}}_X(\underline{\boldsymbol{X}})$ and $R_p(\{p\})$ such that:

$$\begin{aligned}
&\underline{\boldsymbol{\delta u}} = \underline{\boldsymbol{R}}_X(\underline{\boldsymbol{X}})R_p(\{p\}) \qquad (13)\\
&\int_{\Omega^0\times\mathcal{P}} \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{u}}^*,\underline{\boldsymbol{u}}) : (\underset{\approx}{\boldsymbol{\Sigma}}(\underset{\sim}{\boldsymbol{F}}(\underline{\boldsymbol{u}}),\tau\le t,\{p\})\\
&\quad + \underset{\approx}{\boldsymbol{K}} : \underset{\sim}{\boldsymbol{\varepsilon}}(\underline{\boldsymbol{\delta u}},0))\, d\Omega^0\, dp\\
&\quad -\int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^*.\underline{\boldsymbol{f}}(\underline{\boldsymbol{X}},t)\, d\Gamma^0\, dp\\
&=\int_{\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^*.\underline{\widehat{\boldsymbol{r}}}_v\, d\Omega^0\, dp\\
&\quad +\int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\boldsymbol{u}}^*.\underline{\widehat{\boldsymbol{r}}}_b\, d\Gamma^0\, dp \quad \forall\underline{\boldsymbol{u}}^* \in \mathcal{V}_R \qquad (14)
\end{aligned}$$

$$\int_{\Omega^0\times\mathcal{P}} \underline{\widehat{\boldsymbol{r}}}_v.\underline{\widehat{\boldsymbol{r}}}_v\, d\Omega^0\, dp + \int_{\partial_f\Omega^0\times\mathcal{P}} \underline{\widehat{\boldsymbol{r}}}_b.\underline{\widehat{\boldsymbol{r}}}_b\, d\Gamma^0\, dp \le \epsilon_r \qquad (15)$$

where $\underset{\approx}{\boldsymbol{K}}$ is the classical tangent stiffness tensor. As proposed in [11] $\mathcal{V}_R$ is the set of test functions expressed as:

$$\underline{\boldsymbol{u}}^* = \underline{\boldsymbol{R}}_X^*(\underline{\boldsymbol{X}})R_p(\{p\}) + \underline{\boldsymbol{R}}_X(\underline{\boldsymbol{X}})R_p^*(\{p\}) \qquad (16)$$

Due to this choice of test functions the previous problem is a nonlinear problem with respect to $\underline{\boldsymbol{R}}_X$ and R_p. The new functions $\underline{\boldsymbol{\phi}}$ and g are obtained by normalizing the functions $\underline{\boldsymbol{R}}$ and R_p respectively. A convenient approximate solution of (13) to (15) is provided by a fixed point algorithm. The first guess field $\underline{\boldsymbol{R}}_X$ is given by the equilibrium condition related to the point of $\mathcal{P}$ having the highest equilibrium residue. Concerning the numerical implementation of the method, $\underline{\boldsymbol{R}}_X$ is approximated using the classical 3D finite element description. R_p is a piecewise constant function. Therefore the multidimensional simulation can be viewed as a simultaneous simulation of similar mechanical problems. The problem to solve in order to find $\underline{\boldsymbol{R}}_X$ has the same dimension as a classical finite element model related to a unique value of parameters. Due to the separated representation of the displacements and due to the use of the finite element representation of the functions of the variable $\underline{\boldsymbol{X}}$, it is not always possible to choose ϵ_R as small as expected. We refer the reader to [15] for the analysis of the residue of the equilibrium equation depending on the finite element description.

4 Constitutive Law for Sintering Deformation

The proposed separated representation, the truncated integration scheme and the algorithm are generic. They can be applied to various mechanical problems. To illustrate the fact that complexity of the constitutive equations has no consequences on the applicability of the proposed method we applied it to simulate sintering transformations. These transformations can be viewed as irreversible thermal transformations coupled with viscoplastic transformations. The temperature is assumed to be uniform over the multidimensional domain $\Omega^0\times\mathcal{P}\times]t_0,t_f]$. In this paper, the multidimensional simulation aims to estimate the sensitivity of mechanical response to variations of material coefficients.

In general, the strain rate during sintering cycle of a ceramic body, consists of three terms: a reversible thermal strain rate, an irreversible sintering strain rate, and a viscoplastic strain rate, as shown in the following equation [1, 16–18]:

$$\dot{\underset{\sim}{\boldsymbol{\varepsilon}}} = \dot{\varepsilon}_{th}\underset{\sim}{\mathbf{1}} + \dot{\varepsilon}_s(f,T)\underset{\sim}{\mathbf{1}} + \dot{\underset{\sim}{\boldsymbol{\varepsilon}}}_{vp} \qquad (17)$$

where $\underset{\sim}{\mathbf{1}}$ is the 2nd order unit tensor. The strain rate tensor is defined using the polar decomposition of the deformation gradient:

$$\begin{aligned} &\underset{\sim}{\boldsymbol{F}} = \underset{\sim}{\boldsymbol{R}}.\underset{\sim}{\boldsymbol{U}} \quad \text{and} \\ &\dot{\underset{\sim}{\boldsymbol{\varepsilon}}} = \frac{1}{2}\underset{\sim}{\boldsymbol{R}}^T.(\dot{\underset{\sim}{\boldsymbol{F}}}.\underset{\sim}{\boldsymbol{F}}^{-1} + \underset{\sim}{\boldsymbol{F}}^{-T}.\dot{\underset{\sim}{\boldsymbol{F}}}^T).\underset{\sim}{\boldsymbol{R}} \end{aligned} \tag{18}$$

where $\underset{\sim}{\boldsymbol{R}}$ is a pure rotation tensor. The conjugate stress $\underset{\sim}{\boldsymbol{G}}$ is such that:

$$\underset{\sim}{\boldsymbol{\sigma}} = \underset{\sim}{\boldsymbol{R}}.\underset{\sim}{\boldsymbol{G}}.\underset{\sim}{\boldsymbol{R}}^T \tag{19}$$

The sintering strain rate is a function of the temperature (T) and the porosity (f) as expressed in the following term:

$$\dot{\varepsilon}_s(f,T) = -f^y A_s \exp\left(\frac{-Q_s}{RT}\right) \tag{20}$$

The viscoplastic strain rate induced by the mechanical stress $\underset{\sim}{\boldsymbol{G}}$ is described as:

$$\underset{\sim}{\boldsymbol{G}} = \eta(T)\underset{\sim}{\boldsymbol{M}}^{(4)}(T,f) : \dot{\underset{\sim}{\boldsymbol{\varepsilon}}}_{vp} \tag{21}$$

Where η is the viscosity, $\underset{\sim}{\boldsymbol{M}}^{(4)}(T,f)$ is the 4th order tensor which does not generally depend on the stress tensor $\underset{\sim}{\boldsymbol{\sigma}}$ since the stress level is very small:

$$\underset{\sim}{\boldsymbol{M}}^{(4)}(T,f) = F(f)\underset{\sim}{\boldsymbol{I}}^{(4)} + \frac{3}{2}C(f)\underset{\sim}{\boldsymbol{J}}^{(4)} \tag{22}$$

where $\underset{\sim}{\boldsymbol{I}}^{(4)}$ and $\underset{\sim}{\boldsymbol{J}}^{(4)}$ are fourth order tensors that respectively extract the trace and the deviatoric part of a second order tensor:

$$\underset{\sim}{\boldsymbol{I}}^{(4)} : \dot{\underset{\sim}{\boldsymbol{\varepsilon}}}_{vp} = \mathrm{Tr}(\dot{\underset{\sim}{\boldsymbol{\varepsilon}}}_{vp})\underset{\sim}{\mathbf{1}} \tag{23}$$

$$\underset{\sim}{\boldsymbol{J}}^{(4)} : \dot{\underset{\sim}{\boldsymbol{\varepsilon}}}_{vp} = \dot{\underset{\sim}{\boldsymbol{\varepsilon}}}_{vp} - \frac{1}{3}\mathrm{Tr}(\dot{\underset{\sim}{\boldsymbol{\varepsilon}}}_{vp})\underset{\sim}{\mathbf{1}} \tag{24}$$

The coefficients C and F depend on the porosity (f). In a dense body case where $f = 0$, $C = 1$ and $F = 0$. They are assumed as the following form:

$$C(f) = 1 + xf^n, \qquad F(f) = zf^m \tag{25}$$

The viscosity is expressed as:

$$\eta(T) = B_s \exp\left(\frac{-Q_s}{RT}\right) \tag{26}$$

Where Q_s and T are activation energy and absolute temperature respectively. The change of porosity is governed by the strain rate such that:

$$\dot{f} = (1-f)\,\mathrm{Tr}(\dot{\underset{\sim}{\boldsymbol{\varepsilon}}} - \dot{\varepsilon}_{th}\underset{\sim}{\mathbf{1}}) \tag{27}$$

A_s, B_s, Q_s, m, n, x, y and z are the intrinsic coefficients of the material which have to be identified by experimental tests.

5 Numerical Results of the Multidimensional Simulation

5.1 The Finite Element Representation

As mentioned in the previous section, the isotropic sintering deformations consist of various parameters associated with either sintering (thermal) or mechanical solicitations. Due to the variability of the material properties, it is interesting to study the sensitivity of the mechanical response to different material coefficients. One can imagine that the parameters that have the smallest effect could be identified for a large class of materials. Otherwise, the parameters having a greater effect should be specified with greater accuracy for each considered materials.

In this study, the effect of perturbations of mechanical input parameters have been investigated on the bending of a cantilever beam as the response. The response of two results to the input perturbation were investigated: (i) Maximum deflection of the cantilever beam due to sintering (U_2); (ii) Maximum horizontal displacement (U_1) of the cantilever beam during thermal treatment. The column of responses is denoted $\{Y\}$, $\{Y\}^T = \{U_2, U_1\}$. Two levels full factorial design experiment is considered on 5 mechanical parameters of the model. In other words, 2^5 points of $\mathcal{P}$ have been investigated in order to create a Latin hypercube sampling. We propose to simulate simultaneously these 32 numerical cases. The representation of the parameters being piecewise constant, it turns out that the numerical model is a discontinuous finite element model defined in a virtual Euclidean space $\widehat{\Omega}^0$ which mimics the domain $\Omega^0 \times \mathcal{P}$. The mechanical system defined in $\widehat{\Omega}^0$ contains 32 cantilever beams which are associated with every perturbation case studies. This geometry is presented in Fig. 1 which possesses 12800 quadrangular elements (32×400).

The Reduced Integration Domain has been built using the method proposed in [12] and adding all the element of the first beam into the RID. Figure 2 shows the elements in red over which the function h in (10) is not null.

5.2 Parameter Assignation

The parameters whose values were constant throughout the investigation were all associated with sintering (thermal) aspect of the model. These fixed parameters and the values assigned to them are listed in Table 1.

The influence of $\pm 5\%$ perturbation of mechanical parameters of the model is examined using the proposed separated-representation-based algorithm. The corresponding model coefficients and their physical-based optimized values have been presented in Table 2.

The first two parameters (x and n) are associated with the deviatoric part of the model ($C(f)$) which represents the

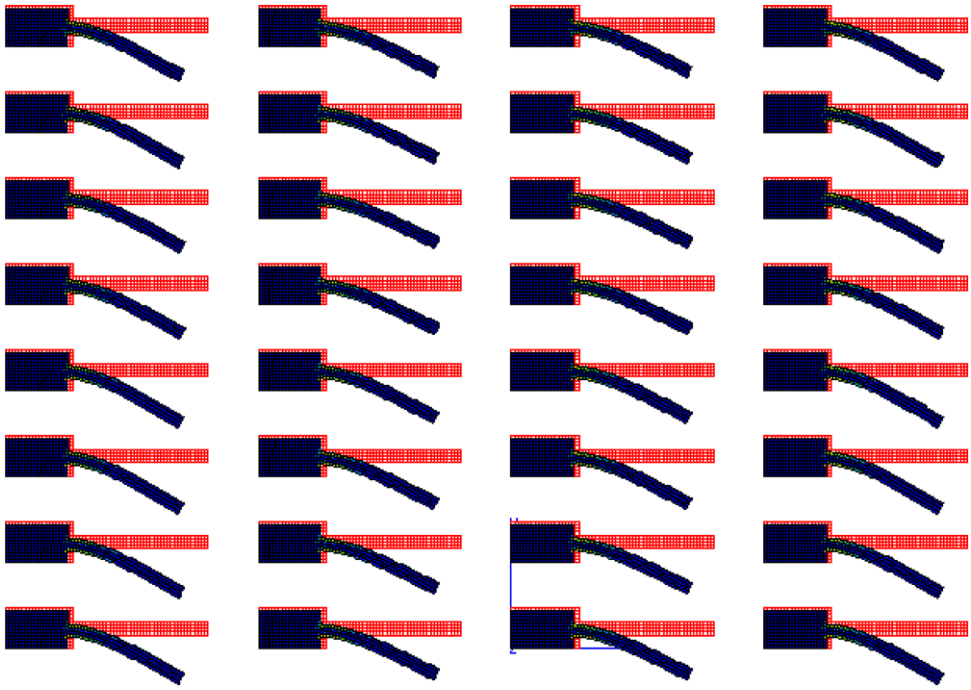


Fig. 1 Mesh and example of Von Mises stress related to the multidimensional problem viewed in virtual 2D space $\widehat{\Omega}^0$ that mimic $\Omega^0 \times \mathcal{P}$

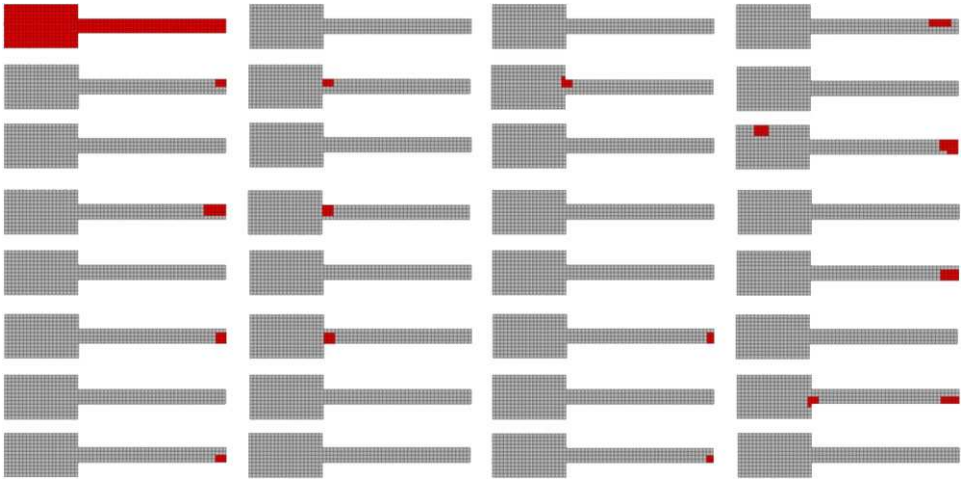


Fig. 2 (Color online) Finite element mesh and reduced integration domain (*red* or *dark gray*) used for mechanical parameters sensitivity analysis with the separated representation

Table 1 Sintering parameters after optimisation

Model parameters	Optimised values
A_S	1.2548274354e+3
y	4.5206134869
Q_S	1.8133327934e+5

Table 2 Mechanical parameters after optimisation

Model parameters	Optimised values
x	1.45386139054
n	1.06450577625
z	9.16276723203e−1
m	1.2735633572
B_S	8.76336201142e−3

distortion due to shear solicitation during sintering. The next couple of input parameters (z and m) which are related to the spherical part of the model ($F(f)$) represent the volume change (shrinkage due to sintering) of the porcelain. The last influential parameter (B_s) is representative of the viscosity of powder made body during thermal treatment.

5.3 Construction of the Sensitivity Matrix

Each of the five variable parameters of the model was assigned high and low values and a unique simulation was run

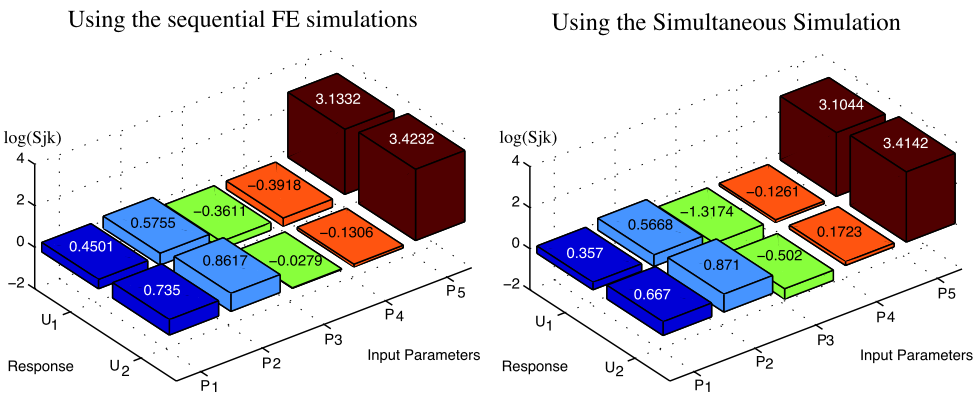


Fig. 3 Sensitivity of response to parameters perturbation in full FEM simulation

with every possible combination of parameter assignments, i.e. a 2^5 samplings:

$$\{p\} = \begin{pmatrix} x \\ n \\ z \\ m \\ B_s \end{pmatrix} = \{p_0\}, \{p_1\}, \{p_2\}, \ldots, \{p_{32}\} \tag{28}$$

After the simultaneous simulation, it is possible to use standard techniques to analyse the variance of the response variables which was performed on a PC using spreadsheet and matlab. We assume that the two responses of interest are linearly linked to the five parameters by a sensitivity matrix $[S]$. The response deviation from the reference case was calculated for all the possible cases by:

$$\{\delta Y_i\} = \{Y_i\} - \{Y_0\} \tag{29}$$

In the same way, the coefficient deviation from the reference value was also determined:

$$\{\delta p_i\} = \{p_i\} - \{p_0\} \tag{30}$$

Then the sensitivity matrix is decomposed into two different matrixes and related to the parameters deviation matrix:

$$\{\delta Y_i\} = [G_i].\{\widehat{S}\} = [S].\{\delta p_i\} \tag{31}$$

The matrix $[G_i]$ involves the values of $\{\delta p_i\}$ in order to introduce a column form of the sensitivity matrix. By minimizing the following term, the non-dimensionalised modulus ($d(\widehat{S})$) used as sensitivity index was obtained:

$$J(\widehat{S}) = \sum_{i=1}^{32} (\{\delta Y_i\} - [G_i].\{\widehat{S}\})^T.(\{\delta Y_i\} - [G_i].\{\widehat{S}\}) \tag{32}$$

$$dJ(\widehat{S}) = -2\sum_{i=1}^{32} d(\widehat{S})^T.[G_i]^T(\{\delta Y_i\} - [G_i].\{\widehat{S}\}) \tag{33}$$

$$dJ(\widehat{S}) = 0 \tag{34}$$

$$\left(\sum_{i=1}^{32}[G_i]^T.[G_i]\right).\widehat{S} = \sum_{i=1}^{32}[G_i]^T.\{\delta Y_i\} \tag{35}$$

5.4 Accuracy of the Simultaneous Simulation Using Truncated Integration Scheme

In this part, the results provided by the proposed algorithm and the results obtained by a classical sequential FE solution are compared. The classical FE solution consists of 32 FE solutions related to each parameter value. This solution being performed one after an other. The sensitivity histogram of full FEM computation is presented in Fig. 3 showing the 10 sensitivity coefficients (5 parameters × 2 responses) of matrix $[S]$. The x axis shows the 5 input parameters (x, n, z, m, B_s) respectively while the responses (U_1 and U_2) are located on y axis. The relative sensitivity of the response variables to the input parameters can be obviously noticed along z axis, using a logarithmic scale. For instance, ±5% perturbation of 5th parameter (B_s) which is related to the viscosity term of the model, is the most influential in terms of the sensitivity magnitude on both responses whereas the same perturbation of parameters associated with the spherical part of the model (z and m) doesn't disturb either of the responses.

As a matter of hierarchy, the deflection of the cantilever beam and its horizontal displacements during sintering are mostly sensitive to viscosity perturbation rather than either the deviatoric or spherical part of the model. Therefore the viscosity should be determined accurately. The other mechanical parameters of the method could be possibly determined for a large set of similar materials.

The qualitative results obtained using a separated-representation-based algorithm are very closed to that of full finite element analysis (Fig. 3) since the most sensitivity is caused by viscosity disturbance (B_s) and neither of responses are disturbed by the parameters of the spherical part of the model (z and m).

Table 3 Global solutions N_g, local solutions N_l and number of unknowns N_u related to each strategy

Strategies	N_g	N_l	N_u
Sequential FEM	$32 \times 981 = 31392$	$32 \times 1{,}569{,}600 = 50{,}227{,}200$	946
Simultaneous simulation without truncated integration scheme	1399	60,108,800	1399
Simultaneous simulation using truncated integration scheme	1399	7,760,400	1399

5.5 Efficiency of the Simultaneous Simulation Using Truncated Integration Scheme

In this section, three computational strategies were compared: (i) the classical FE sequential solution; (ii) the simulation of the simultaneous problems using a separated-representation-based algorithm without truncated integration scheme; (iii) the simulation of the simultaneous problems using both a separated-representation-based algorithm and a truncated integration scheme. As mentioned above, the problem to solve in order to find $\underline{\boldsymbol{R}}_X$ has the same dimension as a classical finite element model related to a unique value of parameters. Therefore, the number of linear problems of classical FE dimension solved when using each strategy is compared. This number is denoted N_g. For this example the dimension of this linear problem is 946. Moreover, to evaluate the complexity reduction due to the use of the truncated integration scheme, we compare the number of local computations (at Gauss points) of the stress over a time increment involved when using each strategy. This number is denoted N_l. Finally, the number of unknowns, denoted N_u, related to each representation were also compared. For the classical FEM strategy N_u is the number of degrees of freedom. For the proposed strategy N_u is the value of γ, (1), at the end of the simultaneous simulation.

As the simultaneous problems are uncoupled, it does not make sense to compare γ to the number of degrees of freedom of the simultaneous FE model which is 32×946. Therefore in this case there is no reduction of the number of unknowns. But the number of linear global solutions has been divided by 22. Without using the proposed truncated integration scheme the separated-representation-based algorithm increases the complexity of the local computations because the related equations are inseparable. The truncated integration scheme enables us to reduce the complexity of the local computations because the number of computations of the stress at a Gauss point over a time increment has been divided by 6.5. Therefore, the truncated integration scheme makes very efficient the solver based on the separated representation in case of simultaneous simulation involving inseparable equations.

6 Conclusion

The sensitivity analysis of mechanical parameter of sintering model has been performed by three different strategies: (i) Full (classic) sequential finite element analysis; (ii) simultaneous simulation using a separated-representation-based solver without truncated integration scheme; (iii) simultaneous simulation using a separated-representation-based solver with truncated integration scheme. The separated representation has been applied distinguishing three kinds of functions: (i) the space functions, (ii) the time functions, (iii) and the functions depending on the material coefficients of the sintering model. The comparison of the results of the three strategies shows the identical sensitivity matrices with different computational complexities. It is clear that the truncated integration scheme improves the efficiency of the separated representation in this case which involves inseparable equations. The number of local computations related to the stress evaluation has been divided by 6.5. During the construction of the separated representation new state functions have been constructed solving linear systems of a dimension equal to the dimension of the classical FE sequential model. The complexity reduction related to the space functions comes from the reduction of the number of linear problems to solve. This number has been divided by 22. Work in progress concerns the reduction of terms involved in the separated representation in case of simultaneous simulation, and the parallelization of the proposed solver.

References

1. Besson J, Abouaf M (1991) Behaviour of cylindrical hip containers. Int J Solids Struct 691–702
2. Ammar A, Mokdad B, Chinesta F, Keunings R (2006) A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. J Non-Newton Fluid Mech 139:153–176
3. Ammar A, Mokdad B, Chinesta F, Keunings R (2007) A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. Part ii: transient simulation using space-time separated representations. J Non-Newton Fluid Mech 144:98–121
4. Chinesta F, Ammar A, Joyot P (2008) The nanometric and micrometric scales of the structure and mechanics of materials revisited: an introduction to the challenges of fully deterministic numerical descriptions. Int J Multiscale Comput Eng 6:191–213

5. Chinesta F, Ammar A, Falco A, Laso M (2007) On the reduction of stochastic kinetic theory models of complex fluids. Model Simul Mater Sci Eng 15:639–652
6. Mokdad B, Pruliere E, Ammar A, Chinesta F (2007) On the simulation of kinetic theory models of complex fluids using the Fokker–Planck approach. Appl Rheol 17:1–14
7. Pruliere E, Ammar A, El Kissi N, Chinesta F (2009) Multiscale modelling of flows involving short fibersuspensions. Arch Comput Methods Eng, State Art Rev 16:1–30
8. Germain P, Nguyen QS, Suquet P (1983) Continuum thermodynamics. J Appl Mech 50:1010–1020
9. Lemaitre J, Chaboche J-L (1985) Mecanique des materiaux solides. Dunod, Paris. English version published by Cambridge University Press, Cambridge, 1st edn (1990)
10. Sansour C, Kollmann FG (1997) On theory and numerics of large viscoplastic deformation. Comput Methods Appl Mech Eng 146:351–369
11. Gonzalez D, Ammar A, Chinesta F, Cueto E (2009) Recent advances on the use of separated representations. Int J Numer Methods Eng
12. Ryckelynck D (2005) A priori hypereduction method: an adaptive approach. Int J Comput Phys 202:346–366
13. Ryckelynck D (2009) Hyper reduction of mechanical models involving internal variables. Int J Numer Methods Eng 77(1):75–89
14. Vanieck P, Jaak J, Featherstone WE (2003) Truncation of spherical convolution integrals with an isotropic kernel. Stud Geophys Geod 47:455–465
15. Babuska I, Rheinbolt WC (1978) A posteriori error estimates for adaptive finite element comptation. Numer Methods Eng 12:1597–1615
16. Song J, Gelin JC, Barrire T, Liu B (2006) Experiments and numerical modelling of solid state sintering for 316l stainless steel components. J Mater Process Technol 352–355
17. Gasik M, Zhang B (2000) A constitutive model and FE simulation for the sintering process of powder compacts. Comput Mater Sci 93–101
18. Bordia RK, Zuo R, Guillon O, Salamone SM, Redel J (2006) Anisotropic constitutive laws for sintering bodies. Acta Mater 111–118

Méthodes numériques de représentation à variables séparées pour la résolution des problèmes paramétriques en mécanique non-linéaire des structures

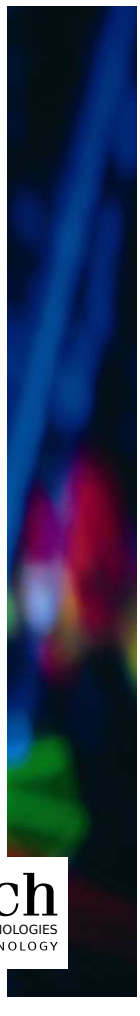
Résumé : Le principal objectif de ce travail est de proposer une méthode de simulation de transformations thermomécaniques bien adaptée aux problèmes d'optimisation traités en milieu industriel ou en laboratoire. Il y a deux types d'approches en optimisation : l'optimisation avec réalisation de suites de simulations thermomécaniques en cours de recherche de l'optimum, ou l'optimisation à l'aide de surfaces de réponses, construites grâce à un ensemble de simulations avant de commencer la recherche de l'optimum. Pour ces deux approches, nous proposons d'exploiter une méthode de réduction adaptative de modèles (APHR), permettant ainsi d'obtenir des modèles simplifiés capables de mieux capter les différentes sensibilités de la réponse du système aux variations des paramètres à optimiser.
La première approche consiste donc à effectuer une suite de calculs en cours d'optimisation. Nous proposons de compléter la méthode APHR par une méthode de gestion des évènements récurrents apparaissant dans différentes prévisions. Le principe de la solution proposée est d'introduire un coefficient d'oubli dans la définition des modes empiriques. Elle a été illustrée sur un problème élasto-plastique avec prévision des dommages par une loi de Rousselier, sur lequel nous avons cherché à recaler les paramètres matériaux. Ce facteur d'oubli a permis d'améliorer l'efficacité de la méthode APHR dans le cadre du recalage de modèle.
Concernant l'optimisation à l'aide de surfaces de réponses, nous nous intéressons uniquement à la construction de ces surfaces de réponses dans le cadre d'une analyse de sensibilité. L'originalité de l'approche développée consiste à développer une méthode numérique de représentation à variables séparées pour la représentation de problèmes paramétriques. Il s'agit de traiter de façon simultanée l'ensemble de problème multidimensionnel. Cette nouvelle approche a été illustrée sur un modèle de frittage et l'efficacité de la méthode a été prouvée par la réduction de la complexité du problème.

Mots clés : Évènements récurrents, Modele d'Ordre Réduits, Problèmes inverses, Suite de calculs, Simulations multidimensionnelles, Simulations simultanées, Méthode APHR

Numerical methods of separated variables representation for the resolution of parametric problems in nonlinear mechanics of structure

Abstract: The main purpose of this work is to propose a simulation method of thermomechanical transformations, well adapted to industrial or laboratory optimization problems. There is two different approaches: optimization by performing series of thermomechanical simulations while seeking the optimum, or optimization by response surface, which are built thanks to a set of simulations, before seeking the optimum. For these two approaches, we propose to exploit an adaptive reduced-order modeling method (APHR) allowing us to obtain simplified models which are sensitive to any model modifications (such as the variation of the parameters to optimize).
The first approach consists in doing series of computations in process of optimization. We propose to complete the APHR method by managing the recurrent events appearing in different predictions. The principle of the proposed solution is to introduce a forgetting factor into the empirical modes definition. It has been illustrated on the identification of material parameters of damage model. The damage mechanisms relies on the Rousselier constitutive law. This forgetting factor enables us to improve the efficiency of the APHR method in the framework of inverse problems.
With regard to optimization by response surface, we are only interested in the building of these response surfaces for sensitivity analysis. The originality of the developed approach lies in developing a numerical method based on separated representation of parametric problems: it's about performing simultaneously a set of multidimensional problems. This new approach has been illustrated by a sensitivity analysis of the response of a sintering model with respect to variation of material coefficients. The efficiency of the method has been proved by reducing the complexity of the problem.

Keywords: Recurrent events, Reduced Order Models, Inverse Problems, Series of computation, Multidimensional Simulations, Simultaneous Simulations, APHR Method



MINES ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY